ISP/Bukavu

Département de Mathématique-Physique

Recherche opérationnelle
ROP1361

Dr. Dieudonné Z. BALIKE, Ph.D.

Cours destiné aux étudiants de 3è Bachelier Math-Info & Math-Physique

2024-2025

Table des matières

Contrat pédagogique

Identification de l'UE

Code : ROP1361
Intitulé : Equations différentielles ordinaires.
Rattachement : Section Sciences Exactes, Département de Mathématique-Physique/Informatique.
Destinataires : Étudiants de 3ème année de Bachelier en Math-Informatique et Math-Physique de l'ISP/Bukavu.
Prérequis : Pour bien participer à ce cours, l'étudiant est supposé avoir des connaissances en Algèbbre linéaire, en programmation, et logique mathématique.

Présentation de l'animateur

Dieudonné Zirhumanana Balike est chargé des CMI et les TD sont assurés cette année par le CT Ambo Amandure.
M. Balike a un doctorat en Mathématiques Appliquées de l'Université de Naples Federico II, en Italie.
Ses recherches portent sur les équations différentielles ordinaires (EDO) et les équations différentielles aux dérivées partielles et leurs applications en Biologie, en Écologie, en Médecine, etc.

Objectifs de l'UE et Compétences à développer

L'enseignement de la Recherche Opérationnelle (RO) en troisième année de licence (L3) vise à doter les étudiants des compétences fondamentales pour modéliser, analyser et résoudre des problèmes d'optimisation complexes rencontrés dans divers domaines scientifiques et industriels.

Objectifs spécifiques

Acquisition des concepts fondamentaux

— Comprendre les principes de base de l'optimisation mathématique
— Maîtriser les différentes classes de problèmes (linéaire, non-linéaire, en nombres entiers)
— Appréhender les notions de fonction objectif, contraintes et variables de décision

Maîtrise des méthodes de résolution

— Implémenter et utiliser l'algorithme du simplexe pour la programmation linéaire

— Appliquer les méthodes de branch-and-bound pour la programmation en nombres entiers
— Utiliser les algorithmes de plus courts chemins (Dijkstra, Bellman-Ford)
— Résoudre des problèmes de flot maximum et de transport

Développement de compétences en modélisation

— Traduire un problème concret en modèle mathématique approprié
— Identifier la classe d'un problème d'optimisation
— Choisir la méthode de résolution adaptée au problème

Renforcement des capacités d'analyse

— Interpréter les résultats obtenus et les solutions optimales
— Analyser la sensibilité des solutions aux variations des paramètres
— Évaluer la complexité algorithmique des méthodes étudiées.

À l'issue du cours, l'étudiant sera capable de :

— Modéliser mathématiquement un problème d'optimisation réel ;
— Choisir et appliquer la méthode de résolution appropriée ;
— Implémenter des algorithmes d'optimisation fondamentaux ;
— Analyser critique les solutions obtenues.

Matériels du cours

Le cours est sur un support électronique et il est gratuitement distribué aux étudiants sous le format électronique. Le cours et les TD sont sur des supports différents. Des références (majoritairement en anglais et des sites internet seront aussi rendus disponibles pour les approfondissements).
Nous utiliserons le tableau et la craie et de temps en temps nous ferons la projection quand c'est nécessaire.

Déroulement de l'UE

L'UE a lieu au sixième semestre.

Mode d'évaluation

Les étudiants seont évalués de deux manières : d'abord par les TD fréquents en cours et les devoirs et enfin par l'examen. Les travaux parfois seront faits en groupes et individuellement.

Avertissements

Malgré mon effort de netoyer ce manuscrit, il est fort possible qu'il subsite des coquilles et des fautes. N'hésitez pas à me les signaler pour les corriger.
Ce cours peut vite devenir très difficile pour les étudiants qui n'ont pas des bases solides des prérequis donnés.

Introduction générale

0.1 Qu'est-ce que la recherche opérationnelle ?

La recherche opérationnelle (RO) vise à l'amélioration du fonctionnement des entreprises et des organismes publics par l'application de l'approche scientifique. Reposant sur l'utilisation de méthodes scientifiques, de techniques spécialisées et des ordinateurs, la RO permet d'obtenir une évaluation quantitative des politiques, stratégies et actions possibles dans le cours des opérations d'une organisation ou d'un système.

La RO est apparue en Grande-Bretagne durant la seconde guerre mondiale, lorsqu'on décida d'employer des méthodes scientifiques pour étudier divers aspects des opérations militaires. Depuis lors, la RO est devenue un élément important du processus de prise de décision dans de nombreux contextes commerciaux, industriels et gouvernementaux, car elle permet d'appréhender de façon systématique la complexité toujours grandissante des problèmes de gestion auxquels est confronté tant le secteur privé que public.
À la suite des succès obtenus dans le domaine militaire durant la seconde guerre mondiale, la RO a été appliquée durant de nombreuses années à des problèmes de nature opérationnelle dans l'industrie, le transport, etc. Depuis une vingtaine d'années, le champ d'application de la RO s'est élargi à des domaines comme l'économie, les finances, le marketing et la planification d'entreprise. Plus récemment, la RO a été utilisée pour la gestion des systèmes de santé et d'éducation, pour la résolution de problèmes environnementaux et dans d'autres domaines d'intérêt public. Parmi les sujets d'application récents de la RO, on peut mentionner :

— les études logistiques (forces armées),
— la sécurité ferroviaire,
— la conception d'emballages,
— la gestion prévisionnelle du personnel,
— le transport aérien,
— les opérations forestières,
— l'optimisation du carburant nucléaire,
— l'affectation des ressources dans un hôpital,
— l'étude des réseaux de commutation,
— la planification de la production,
— l'apprentissage artificiel,
— etc.

0.2 Une carrière en recherche opérationnelle

Pour réussir, le chercheur opérationnel doit faire preuve de grandes habilités analytiques, d'un esprit ouvert et d'un intérêt marqué pour la résolution de problèmes pratiques.
À l'heure actuelle, on retrouve des personnes possédant une formation en RO à l'intérieur d'équipes spécialisées en RO uvrant pour certains dans divers secteurs d'organismes privés ou publics. Les méthodes de la RO sont aussi appliquées par des économistes, des ingénieurs, des scientifiques, des administrateurs et des cadres supérieurs dans la résolution de problèmes de gestion et de politique d'entreprise.
Depuis l'apparition de la RO, les décideurs et les gestionnaires y ont recours très fréquemment. Ses applications sont en pleine expansion alors que l'envergure et la complexité des problèmes soumis aux praticiens de la RO n'ont pas cessé de s'accroître. En conséquence, le développement de techniques et méthodes nouvelles pour faire face à ces nouveaux problèmes a provoqué chez les gestionnaires de tous les secteurs des affaires, de l'industrie et du gouvernement une prise de conscience sans cesse grandissante de la nécessité de telles techniques et méthodes ainsi qu'une plus grande confiance en ce que la RO peut faire pour la gestion et la prise de décisions. Il y a dans les secteurs public et privé une demande croissante et un besoin certain des services de la RO.

Chapitre

1

Programmation linéaire

1.1 La programmation linéaire - Méthode graphique

1.1.1 Introduction

La programmation mathématique recouvre un ensemble de techniques d'optimisation sous contraintes qui permettent de déterminer dans quelles conditions on peut rendre maximum ou minimum une fonction objectif $Z(X_j)$ de n variables X_j liées par m relations ou contraintes $H_i(X_j) \leq 0$.
De nombreux problèmes de l'entreprise peuvent s'exprimer en termes d'optimisation contrainte, aussi rencontre-t-on de multiples applications de la programmation mathématique et ceci dans pratiquement tous les domaines de la gestion.
La gestion de production est le domaine où ces applications sont les plus nombreuses. On citera entre-autres :

- l'élaboration de plans de production et de stockage,
- le choix de techniques de production,
- l'affectation de moyens de production,
- la détermination de la composition de produits.

Les applications sont également nombreuses dans le domaine du marketing avec, en particulier :

- le choix de plans-média,
- la détermination de politiques de prix,
- la répartition des efforts de la force de vente,
- la sélection des caractéristiques du produit.

On citera encore des applications en matière financière (choix de programmes d'investissements), en matière logistique (gestion des transports) et en matière de gestion des ressources humaines (affectation de personnel).
Si les applications de la programmation mathématique sont aussi nombreuses, on doit l'attribuer en

grande partie à la souplesse de ses techniques en ce qui concerne leur formulation mais aussi à la relative simplicité des méthodes de résolution utilisables dans les cas les plus courants et pour lesquelles existent des programmes informatiques largement répandus. Parmi les techniques de programmation mathématique la programmation linéaire est la plus classique.

1.1.2 Modélisation d'un programme linéaire

La formalisation d'un programme est une tâche délicate mais essentielle car elle conditionne la découverte ultérieure de la bonne solution. Elle comporte les mêmes phases quelles que soient les techniques requises ultérieurement pour le traitement (programmation linéaire ou programmation non linéaire) :

1. La détection du problème et l'identification des variables. Ces variables doivent correspondre exactement aux préoccupations du responsable de la décision. En programmation mathématique, les variables sont des variables décisionnelles.
2. La formulation de la fonction économique (ou fonction objectif) traduisant les préférences du décideur exprimées sous la forme d'une fonction des variables identifiées.
3. La formulation des contraintes. Il est bien rare qu'un responsable dispose de toute liberté d'action. Le plus souvent il existe des limites à ne pas dépasser qui revêtent la forme d'équations ou d'inéquations mathématiques.

Le responsable d'une décision ne dispose que de sa compétence pour réaliser une formalisation correcte du problème posé car il n'existe pas de méthode en la matière. Un moyen d'acquérir cette compétence est l'apprentissage comme proposé dans les exemples suivants :

1.1.3 Exemples

Exemple 1.1.1 *Une usine fabrique deux produits P_1 et P_2 à l'aide de trois matières premières M_1, M_2 et M_3 dont on dispose en quantité limitée. On se pose le problème de l'utilisation optimale de ce stock de matières premières c'est-à-dire la détermination d'un schéma, d'un programme de fabrication tel que :*

— *les contraintes de ressources en matières premières soient respectées,*

— *le bénéfice réalisé par la vente de la production soit maximum.*

Modèle mathématique :

— Données numériques des contraintes. La disponibilité en matières premières est de 18 unités de $M_1, 8$ unités de M_2 et 14 unités de M_3.

— Caractéristiques de fabrication. Elles sont données dans le tableau ci-dessous :

	M_1	M_2	M_3
P_1	1	1	2
P_2	3	1	1

— Hypothèses de linéarité du modèle. La fabrication est à rendement constant, c'est-à-dire que pour fabriquer x_1 unités de P_1, il faut $1 \times x_1$ unités de $M_1, 1 \times x_1$ unités de M_2 et $2 \times x_1$ unités de M_3, de même pour la fabrication de x_2 unités de P_2.

— Linéarité de la fonction économique. On suppose que le bénéfice peut s'exprimer à l'aide des bénéfices unitaires c_1, c_2 sous la forme :

$$Z(x_1, x_2) = c_1x_1 + c_2x_2$$

— Réalisation d'un schéma de production. Un schéma de production est un couple (x_1, x_2), x_1 et x_2 désignant respectivement les quantités de P_1 et P_2 fabriquées donc vendues, qui doit vérifier les contraintes $x_1 \geq 0, x_2 \geq 0$. Deux questions se posent : un tel schéma est-il réalisable ? A-t-on suffisamment de matières premières pour assurer une telle production ?

— *Le programme linéaire :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 3x_2 \leq 18 \\ x_1 + x_2 \leq 8 \\ 2x_1 + x_2 \leq 14 \\ Z(x_1, x_2) = c_1x_1 + c_2x_2 \end{cases}$$

où Z est une fonction économique ou fonction objectif qu'il faut maximiser.

Exemple 1.1.2 *L'intendant d'un lycée doit composer un menu qui doit contenir un minimum d'éléments nutritifs et qui doit être le moins coûteux possible. On se limite à une situation simple, deux denrées alimentaires principales D_1, D_2 et trois éléments nutritifs, les vitamines V, les calories C et les protéines P.*

Le tableau suivant indique le nombre d'éléments nutritifs par unité d'aliment :

	V	C	P
D_1	1	1	3
D_2	5	2	2

Une unité de D_1 contient 1 unité de V, 1 unité de C et 3 unités de P.

Modèle mathématique :

— Contraintes diététiques. Le menu doit comporter au minimum 5 unités de V, 4 unités de C, 6 unités de P . Les coûts unitaires sont 20 pour D_1, 25 pour D_2.

— Réalisation du menu. Un menu contenant x_1 unités de D_1, x_2 unités de D_2 est réalisable si le couple (x_1, x_2) vérifie :

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 5x_2 \geq 5 \\ x_1 + 2x_2 \geq 4 \\ 3x_1 + x_2 \geq 6 \end{cases}$$

— Le programme linéaire. Le problème consiste à déterminer deux nombres x_1 et x_2 tels que :

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 5x_2 \geq 5 \\ x_1 + 2x_2 \geq 4 \\ 3x_1 + x_2 \geq 6 \\ Z(x_1, x_2) = 20x_1 + 25x_2 \end{cases}$$

où Z est la fonction objectif à minimiser.

1.1.4 Formule générale d'un programme linéaire

De façon générale, un problème de programmation mathématique met en jeu quatre catégories d'éléments :

— des variables ou activités,

— des coefficients économiques,

— des ressources,

— des coefficients techniques.

Les activités sont les variables de décision du problème étudié. Il s'agit pour l'entreprise de sélectionner le meilleur programme d'activités $X = (x_1, \ldots, x_n)$, c'est-à-dire celui qui est le plus conforme à ses objectifs. Les coefficients économiques mesurent le degré de réalisation de l'objectif de l'entreprise, associé à une valeur unitaire de chacune des variables. À chaque variable x_j est ainsi associé un coefficient économique c_j. L'évaluation des coefficients c_j dépend du type d'objectif poursuivi : selon le cas ce sera un prix de vente, une marge brute, un coût variable unitaire, etc.

Les ressources peuvent être également de nature très diverse selon le problème rencontré. Dans tous les cas, ce sont les éléments qui limitent le calcul économique de l'entreprise : des capacités de production limitées, des normes à respecter, des potentiels de vente, etc. Dans tout problème, il faudra ainsi prendre en considèration un vecteur de ressources $B = (b_1, \ldots, b_m)$ donné.

Par coefficient technique on désignera le degré de consommation d'une ressource par une activité. À la ressource i et à l'activité j correspondra le coefficient technique a_{ij}. Dans la mesure où le problème étudié met en jeu n activités et m ressources, il faudra considérer $m \times n$ coefficients techniques que l'on pourra regrouper dans un tableau du type suivant :

Activités / Ressources	1	2	$\cdots$	n
1	a_{11}	a_{12}	$\cdots$	a_{1n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	a_{i1}	a_{i2}	$\cdots$	a_{in}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
m	a_{m1}	a_{m2}	$\cdots$	a_{mn}

Si les variables sont continues, si les coefficients économiques et techniques sont indépendants des valeurs des variables, alors le problème peut être formalisé à l'aide d'un programme linéaire.

Un même programme peut être traduit sous une forme canonique ou sous une forme standard ; l'une et l'autre pouvant adopter soit la notation algébrique classique soit la notation matricielle que l'on ne traitera pas ici. Voyons tout d'abord la forme canonique. Elle se caractérise par des contraintes présentées sous la forme d'inéquations telles que

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, \ldots, x_n \geq 0 \\ a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq \text{ ou } \geq \text{ ou } = b_1 \\ \quad \vdots \\ a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq \text{ ou } \geq \text{ ou } = b_i \\ \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq \text{ ou } \geq \text{ ou } = b_m \end{cases} \tag{1.1}$$

et par une forme linéaire

$$Z(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n. \tag{1.1}$$

Résoudre le programme linéaire consiste à déterminer les n-uplets $(x_1, x_2, \ldots, x_n)$ qui optimisent Z (maximisent ou minimisent) Z ou à montrer que de tels n-uplets n'existent pas.
On se donne les définitions suivantes :

Définition 1.1.3 — *On appelle solution réalisable tout n-uplet $(x_1, x_2, \ldots, x_n)$ vérifiant le système d'inéquations précédent.*

— *On appelle solution optimale toute solution réalisable qui optimise Z.*

— *On appelle fonction objectif la forme linéaire*

$$Z(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n$$

— *L'ensemble des solutions réalisables du programme linéaire P est appelé domaine des solutions réalisables. Lorsque ce domaine est non vide, on dit que P est réalisable.*

Résoudre un programme linéaire consiste à déterminer les valeurs des variables qui permettent d'optimiser la fonction économique.
Il existe diverses techniques de résolution parmi lesquelles la méthode graphique se montre à l'évidence la plus rapide et la plus simple mais aussi la plus limitée, car dès lors que le nombre de variables ou de contraintes dépasse 2, elle devient impraticable. C'est pourquoi divers chercheurs se sont efforcés de mettre au point une méthode de calcul algorithmique qui permet de détecter la solution optimale (si elle existe) quel que soit le nombre des variables et des contraintes.
Bien que très efficace, cette méthode connue sous le nom d'algorithme du simplexe, exige des calculs longs et fastidieux. C'est pourquoi ceux-ci sont de plus en plus confiés à l'outil informatique. Dès lors une question se pose : puisque les logiciels correspondants sont largement répandus, est-il nécessaire pour appliquer la méthode, d'en connaître les ressorts ? Deux raisons essentielles justifient une réponse affirmative :

— d'abord, la compréhension des principes de résolution est une aide précieuse pour, en amont, analyser et formaliser le problème et pour, en aval, interpréter et exploiter la solution obtenue ;

— ensuite parce que la démarche algorithmique présente en elle-même un intérêt formateur non négligeable.

1.1.5 Méthode graphique : problème à deux inconnues

1.1.5.1 Régionnement du plan

Le régionnement du plan revient à étudier le signe de $ax + by + c$ avec $(a, b) \neq (0, 0)$.
Si on considère la droite D dont une équation est $ax + by + c = 0$ avec $a \neq 0$ ou $b \neq 0$, cette droite partage le plan en deux demi-plans (I) et (II) de frontière D :

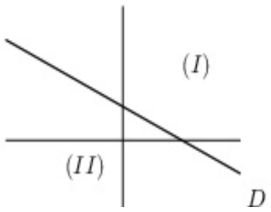


FIGURE 1.1 – Regionnement du plan

— Pour tout point $M(x, y)$ situé sur D, on a $ax + by + c = 0$.

— Pour toute droite D d'équation $ax + by + c = 0$ partitionnant le plan en deux demi-plans (I) et (II), l'expression $ax + by + c$ garde un signe constant dans chaque région. Plus précisément, si pour un point $M(x, y)$ du demi-plan (I) on a $ax + by + c > 0$ (respectivement < 0), alors pour tout point $N(x, y)$ appartenant à l'autre demi-plan (II), on aura nécessairement $ax+by+c < 0$ (respectivement > 0).

Exemple 1.1.4 — *Signe de $x + y - 1$:*

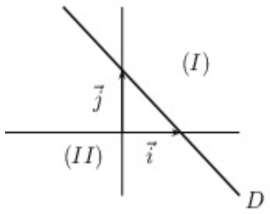


FIGURE 1.2 – Droite d'équation $x + y - 1 = 0$.

On trace la droite d'équation $x+y-1 = 0$. À l'origine, $x+y-1 = (0)+(0)-1 = -1 < 0$ donc pour tous les points $M(x, y)$ situés dans le demi-plan (II), $x + y - 1 < 0$ et pour tous les points $N(x, y)$ situés dans le demi-plan $(I), x+y-1 > 0$. Pour les points $P(x, y)$ de la droite $D, x+y-1$ prend la valeur 0.

— *Signe de $-x + y$:*

On trace la droite D d'équation $-x + y = 0$, cette droite contient l'origine du repère. Pour le point $A(1, 0), x - y = 1 > 0$ donc pour tous les points $M(x, y)$ situés dans le demi-plan $(I), x - y > 0$ et pour tous les points $N(x, y)$ situés dans le demi-plan $(II), x - y < 0$. Pour les points $P(x, y)$ de la droie $D, x - y$ prend la valeur 0 .

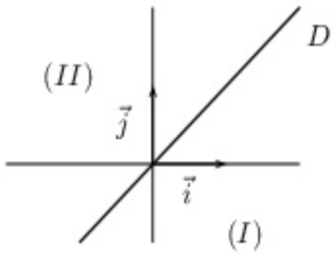


FIGURE 1.3 – Droite d'équation $-x + y = 0$

1.1.5.2 Les ensembles convexes

Définition 1.1.5 *Un ensemble E est dit convexe si pour M_1 et M_2 deux points quelconques de E, tous les points du segment $[M_1, M_2]$ appartiennent à E (Illustrations dans les figures).*
*Un **ensemble convexe** $C \subset \mathbb{R}^n$ est défini par la propriété suivante : pour tout couple de points $x, y \in C$ et pour tout $\lambda \in [0, 1]$, le point*

$$\lambda x + (1 - \lambda)y \in C.$$

Autrement dit, tout segment joignant deux points de C est entièrement contenu dans C.

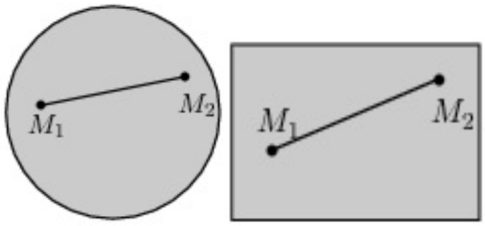


FIGURE 1.4 – Le disque et le rectangle sont des ensembles convexes

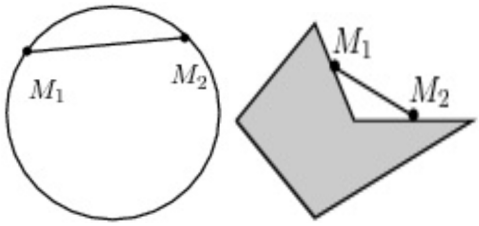


FIGURE 1.5 – Le cercle n'est pas un ensemble convexe : les points du segment $]M_1, M_2[$ n'appartiennent pas au cercle. De même ce polygone n'est pas convexe.

1.1.5.3 Résolution de systèmes d'inéquations - Exemples

Exemple 1.1.6 *On considère le système suivant :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ -x_1 - x_2 \leq -1 \\ x_1 + 4x_2 \leq 2 \\ 6x_1 + x_2 \leq 2. \end{cases} \tag{1.2}$$

Comme $x_1 \geq 0$ et $x_2 \geq 0$, les points $M(x_1, x_2)$ seront choisis dans le quart du plan (voir figure 1.6) : L'ensemble des solutions est représenté par la surface grise.

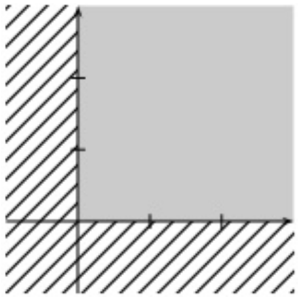
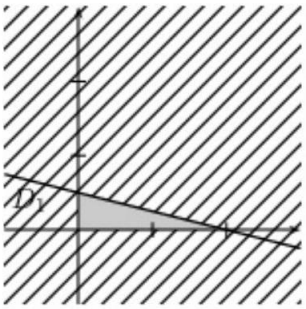


FIGURE 1.6 – La surface grise est l'ensemble des solutions

On considère ensuite le système partiel

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 4x_2 \leq 2 \end{cases}$$

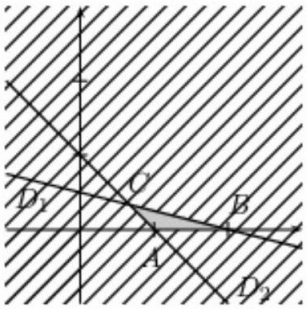
On trace la droite D_1 d'équation $x_1 + 4x_2 = 2$. Comment déterminer le demi-plan qui convient ? Il suffit de prendre un point quelconque du plan et d'observer si ses coordonnées vérifient l'inéquation. Si c'est le cas, le point se situe dans le bon demi-plan. Considérons par exemple l'origine, $x_1 + 4x_2 =$



$0 + 4 \times 0 = 0 \leq 2$ donc l'origine est solution et tous les points situés dans le demi-plan contenant l'origine sont solutions. On considère ensuite le système

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 4x_2 \leq 2 \\ -x_1 - x_2 \leq -1 \end{cases}$$

On trace la droite D_2 d'équation $-x_1 - x_2 = -1$. Considérons l'origine, $-x_1 - x_2 = 0 - 0 = 0 > -1$ donc l'origine n'est pas solution, les solutions du système sont par conséquent les points du triangle ABC et son intérieur avec $A(1,0), B(2,0)$ et $C\left(\frac{2}{3}, \frac{1}{3}\right)$.
On considère enfin le système de départ



$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 4x_2 \leq 2 \\ -x_1 - x_2 \leq -1 \\ 6x_1 + x_2 \leq 2 \end{cases}$$

On trace la droite D_3 d'équation $6x_1 + x_2 = 2$. Considérons le point origine, $6x_1 + x_2 = 6 \times 0 + 0 = 0 < 2$ donc l'origine est solution de l'inéquation. On sélectionne le demi-plan qui convient et on observe finalement que le système n'admet pas de solution (la partie grise est inexistante).

Exemple 1.1.7 *On considère le système suivant :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + x_2 \leq 1 \\ -3x_1 + x_2 \leq -3 \end{cases} \tag{1.3}$$

On sélectionne l'intersection des deux demi-plans $x_1 \geq 0$ et $x_2 \geq 0$.
On considère la droite d'équation $D_1 : x_1 + x_2 = 1$. Le demi-plan qui convient est repéré grâce, par

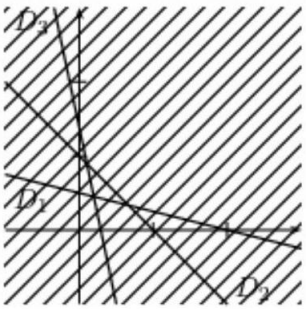


FIGURE 1.7 – Le système d'inéquation (1.2) n'admet pas de solution (pas de zone grise).

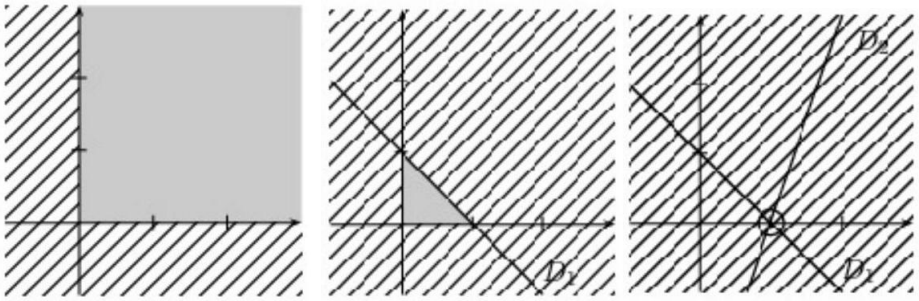


FIGURE 1.8 – Etapes de la résolution du système (1.3).

exemple, à l'origine.
On considère la droite d'équation $D_2 : -3x_1 + x_2 = -3$. Le demi-plan qui convient est repéré une fois de plus grâce à l'origine. L'ensemble solution se restreint à un seul point, le couple solution $(1, 0)$.

Exemple 1.1.8 *On considère le système suivant :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 5x_2 \geq 5 \\ x_1 + 2x_2 \geq 4 \\ 3x_1 + 2x_2 \geq 6 \end{cases} \tag{1.4}$$

Comme $x_1 \geq 0$ et $x_2 \geq 0$, les points $M(x_1, x_2)$ seront choisis dans le quart du plan (premier graphique de la figure 1.9).
On considère la droite d'équation $D_1 : x_1 + 5x_2 = 5$. Le demi-plan qui convient est repéré grâce, par exemple, à l'origine.
On considère la droite d'équation $D_2 : x_1 + 2x_2 = 4$. Le demi-plan qui convient est repéré grâce, par exemple, à l'origine.
On considère la droite d'équation $D_3 : 3x_1 + 2x_2 = 6$. Le demi-plan qui convient est repéré grâce, par exemple, à l'origine.

Exemple 1.1.9 *On considère le système suivant :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 3x_2 \leq 18 \\ x_1 + x_2 \leq 8 \\ 2x_1 + x_2 \leq 14 \end{cases} \tag{1.5}$$

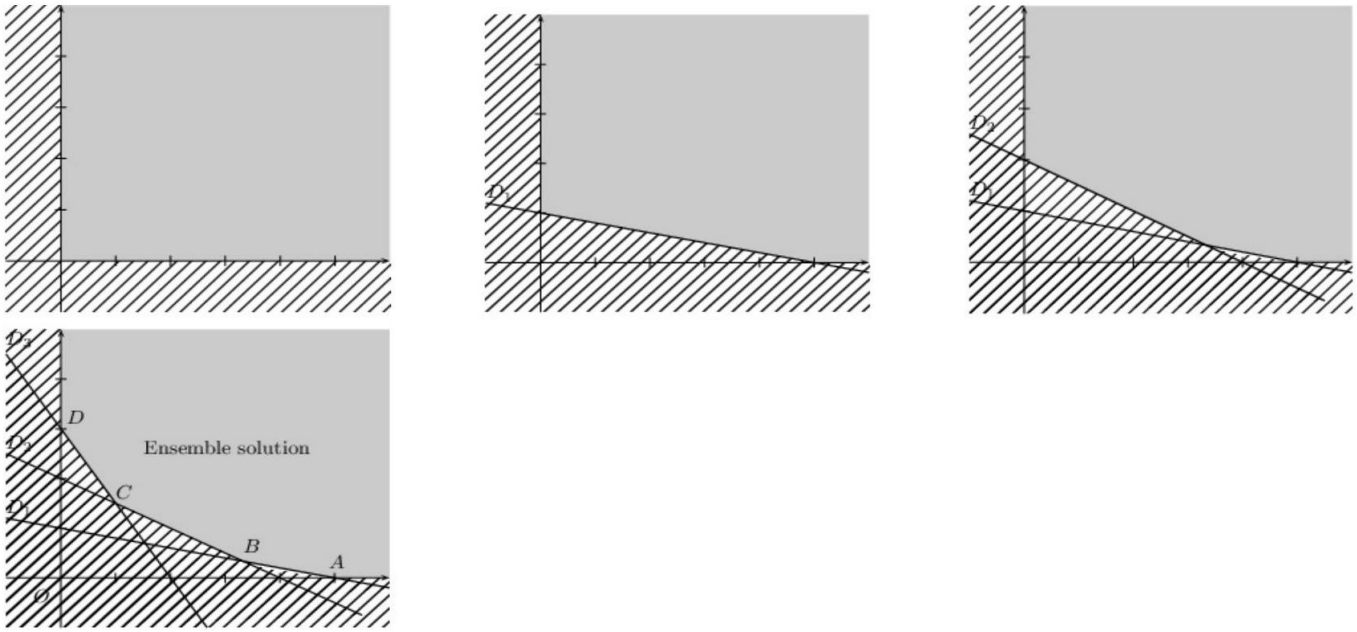
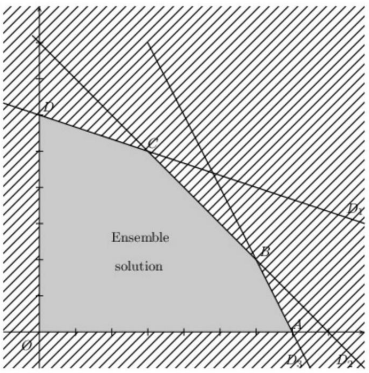


FIGURE 1.9 – Etapes de la résolution du système (1.4).

Soient les droites d'équations respectives

$$D_1 : x_1 + 3x_2 = 18, D_2 : x_1 + x_2 = 8 \text{ et } D_3 : 2x_1 + x_2 = 14.$$

L'ensemble solution est un polyèdre convexe limité par la ligne polygonale $OABCD$.

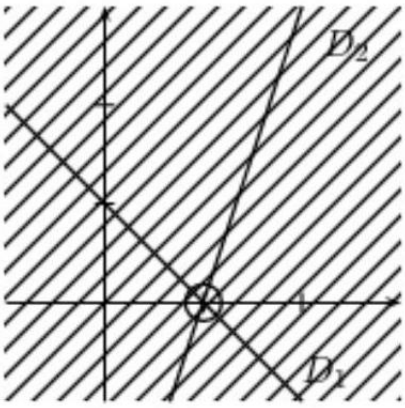


1.1.5.4 Résolution de programmes linéaires

Exemple 1.1.10 *On reprend le système de l'exemple* (1.3) *auquel on ajoute une fonction objectif :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ -3x_1 + x_2 \leq -3 \\ x_1 + x_2 \leq 1 \\ Z(x_1, x_2) = 3x_1 + x_2 \text{ à maximiser} \end{cases} \tag{1.6}$$

On rappelle que le domaine des solutions réalisables est donné graphiquement par : Le programme

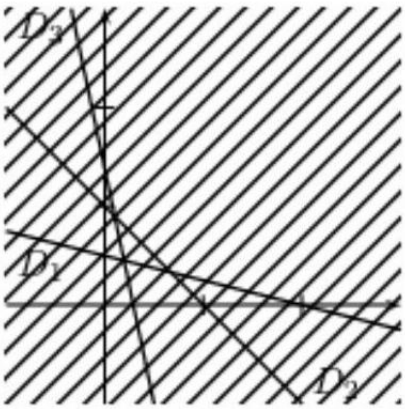


linéaire admet une unique solution réalisable $(1,0)$ qui est d'ailleurs la solution optimale. Z est maximum pour le couple $(1,0)$ et vaut $Z(1,0) = 3 \times 1 + 0 = 3$.

Exemple 1.1.11 *On reprend le système de l'exemple* (1.2) *auquel on ajoute une fonction objectif :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 4x_2 \leq 2 \\ -x_1 - x_2 \leq -1 \\ 6x_1 + x_2 \leq 2 \\ Z(x_1, x_2) = 6x_1 + x_2 \ \textit{à maximiser} \end{cases} \tag{1.7}$$

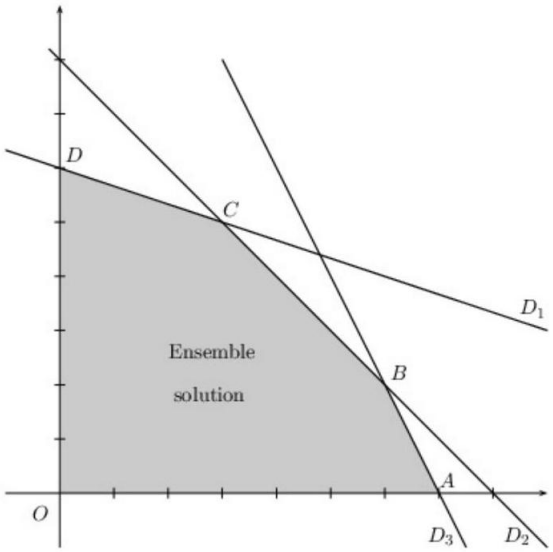
L'ensemble solution est donné graphiquement par :
Ce programme n'a pas de solution réalisable. Le domaine des solutions réalisables est le vide.



Exemple 1.1.12 *On reprend le système de l'exemple* (1.5) *auquel on ajoute une fonction objectif :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 3x_2 \leq 18 \\ x_1 + x_2 \leq 8 \\ 2x_1 + x_2 \leq 14 \\ Z(x_1, x_2) = 2x_1 + 4x_2 \ \textit{à maximiser} \end{cases} \tag{1.8}$$

Le domaine des solutions réalisables est donné graphiquement par :
Le domaine des solutions réalisables est un domaine plan, délimité par le polygone $OABCD$. Le

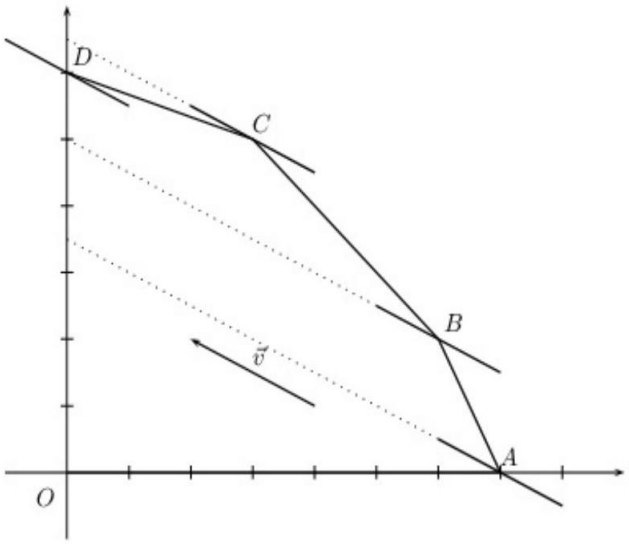


domaine plan est un ensemble convexe. On détermine ensuite les couples (x_1, x_2) de solutions réalisables tels que $Z(x_1, x_2) = 2x_1 + 4x_2$ soit maximum. Pour tout nombre Z, on note D_Z la droite d'équation

$$Z = 2x_1 + 4x_2$$

appelée généralement droite d'isovaleur de la fonction objectif. Un vecteur directeur de cette droite D_Z est $\vec{v}\binom{-4}{2}$ ou $\vec{w}\binom{-2}{1}$. Son coefficient directeur est $-\frac{1}{2}$. En effet, $x_2 = -\frac{1}{2}x_1 + \frac{Z}{4}$. Lorsque Z varie, ces droites D_Z ayant même coefficient directeur sont parallèles entre elles. L'ordonnée à l'origine des droites D_Z est $\frac{Z}{4}$. Maximiser Z est équivalent à maximiser $\frac{Z}{4}$. Le problème consiste donc à déterminer une ou plusieurs droites D_Z qui rencontrent le domaine des solutions réalisables et ayant une ordonnée à l'origine maximale. Lorsque Z augmente, la droite D_Z se déplace parallèlement à elle même vers le haut :

La droite D_Z qui rencontre le domaine des solutions réalisables et qui a une ordonnée à l'origine



maximale est celle qui contient le point C.
Le programme linéaire a une seule solution maximale, le couple $(3, 5)$.

En conclusion, pour $x_1 = 3, x_2 = 5$, la fonction objectif est maximale et vaut

$$Z(3,5) = 2 \times 3 + 4 \times 5 = 26$$

Remarque 1.1.13 *La fonction objectif atteint son maximum en un des sommets du polygone.*

Exemple 1.1.14 *On considère le système*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + x_2 \geq 2 \\ 2x_1 + x_2 \geq 3 \\ Z(x_1, x_2) = -x_1 + x_2 \ \textit{à minimiser} \end{cases} \tag{1.9}$$

Le domaine des solutions réalisables est donné graphiquement par la figure 1.10 : Le domaine des

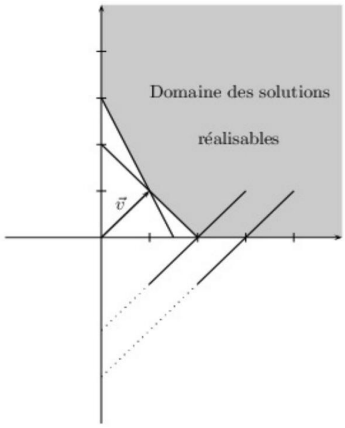


FIGURE 1.10 – Ensemble de solutions réalisables du système (1.9)

solutions réalisables est convexe. Minimisons la fonction objectif : pour Z donné, on trace la droite D_Z d'équation $-x_1 + x_2 = Z \Leftrightarrow x_2 = x_1 + Z$. Lorsque Z varie, ces droites D_Z de vecteur directeur $\binom{1}{1}$ (de coefficient directeur 1) sont parallèles entre elles. On recherche une ou plusieurs droites D_Z ayant une ordonnée à l'origine Z minimale. Pour toute valeur de $Z(\in \mathbb{R}), D_Z$ rencontre le domaine des solutions réalisables. Le programme linéaire n'a pas de solution minimale.

Exemple 1.1.15 *On reprend le système de l'exemple* (1.4) *auquel on ajoute une fonction objectif :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 5x_2 \geq 5 \\ x_1 + 2x_2 \geq 4 \\ 3x_1 + 2x_2 \geq 6 \\ Z(x_1, x_2) = 20x_1 + 25x_2 \ \textit{à minimiser} \end{cases} \tag{1.10}$$

Le domaine des solutions réalisables est donné graphiquement par 1.11. Pour Z donné, on trace la droite D_Z d'équation $Z(x_1, x_2) = 20x_1 + 25x_2$ ou encore $x_2 = -\frac{4}{5}x_1 + \frac{Z}{25}$. Cette droite D_Z a pour coefficient directeur $-\frac{4}{5}$, pour vecteur directeur $\vec{v}\binom{-25}{20}$ ou $\vec{w}\binom{-5}{4}$ et pour ordonnée à l'origine $\frac{Z}{25}$. On trace des droites D_Z de coefficient directeur $-\frac{4}{5}$ et on recherche une ou plusieurs droites D_Z, rencontrant le domaine des solutions réalisables et ayant une ordonnée à l'origine $\frac{Z}{25}$ minimale. La droite D_Z rencontrant le domaine des solutions réalisables et ayant une ordonnée à l'origine minimale est celle qui contient le point $C\left(1, \frac{3}{2}\right)$. La fonction objectif atteint son minimum pour le couple $\left(1, \frac{3}{2}\right)$ et vaut $Z\left(1, \frac{3}{2}\right) = 20 \times 1 + 25 \times \frac{3}{2} = \frac{115}{2}$.

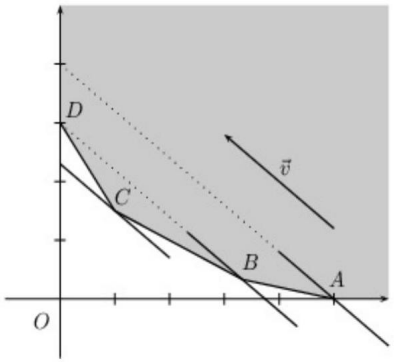


FIGURE 1.11 – Ensemble de solutions réalisables du système (1.10)

Exemple 1.1.16 *On considère le système mis en place dans le cadre de l'exemple* (1.5) *:*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 + 3x_2 \leq 18 \\ x_1 + x_2 \leq 8 \\ 2x_1 + x_2 \leq 14 \\ Z(x_1, x_2) = c_1 x_1 + c_2 x_2 \end{cases} \tag{1.11}$$

où Z est une fonction économique ou fonction objectif qu'il faut maximiser et c_1 et c_2 sont les bénéfices unitaires.
Résolvons ce problème linéaire, on discutera bien-sûr des valeurs attribuées à c_1 et c_2.
Le domaine des solutions réalisables est le domaine convexe délimité par le polygone $OABCD$. Les coordonnées des sommets sont obtenues en déterminant les intersections des droites donc en résolvant des systèmes de deux équations à deux inconnues.

Étude de cas particuliers

— $c_1 = 1, c_2 = 4$: on trace les droites D_Z d'équations :

$$x_1 + 4x_2 = Z \Leftrightarrow x_2 = -\frac{1}{4}x_1 + \frac{Z}{4}$$

de vecteur directeur $\overrightarrow{v_1}\binom{-4}{1}$. La droite qui a une ordonnée à l'origine maximale est celle qui contient le point $D\binom{0}{6}$. La fonction objectif est maximale pour le couple $(0, 6)$ et vaut $Z(0, 6) = 0 + 4 \times 6 = 24$.

— $c_1 = 2, c_2 = 4$: on trace les droites D_Z d'équations :

$$2x_1 + 4x_2 = Z \Leftrightarrow x_2 = -\frac{1}{2}x_1 + \frac{Z}{4}$$

de vecteur directeur $\overrightarrow{v_2}\binom{-2}{1}$. La droite qui a une ordonnée à l'origine maximale est celle qui contient le point $C\binom{3}{5}$. La fonction objectif atteint son maximum au point $(3, 5)$ et vaut $Z(3, 5) = 2 \times 3 + 4 \times 5 = 26$.

— $c_1 = 2, c_2 = 2$: on trace les droites D_Z d'équations :

$$2x_1 + 2x_2 = Z \Leftrightarrow x_2 = -x_1 + \frac{Z}{2}$$

de vecteur directeur $\overrightarrow{v_3}\binom{-1}{1}$. Cette droite D_Z est parallèle au côté (BC) du polygone. La fonction objectif atteint son maximum en tous les points du côté (BC). La fonction objectif atteint donc ce maximum pour tous les couples (x_1, x_2) tels que $x_1 + x_2 = 8$ et $3 \leq x_1 \leq 6$. Z vaut alors $2x_1 + 2x_2 = 16$.

— $c_1 = 3, c_2 = 2$: on trace les droites D_Z d'équations :

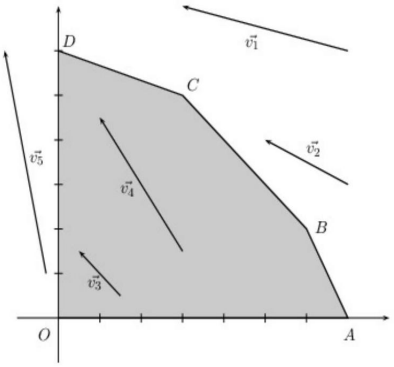
$$3x_1 + 2x_2 = Z \Leftrightarrow x_2 = -\frac{3}{2}x_1 + \frac{Z}{2}$$

de vecteur directeur $\overrightarrow{v_4}\binom{-2}{3}$. La droite qui a une ordonnée à l'origine maximale est celle qui contient le point $B\binom{6}{2}$. La fonction objectif atteint son maximum au point (6,2) et vaut $Z(6,2) = 3\times 6 + 2\times 2 = 22$.

— $c_1 = 5, c_2 = 1$: on trace les droites D_Z d'équations :

$$5x_1 + x_2 = Z \Leftrightarrow x_2 = -5x_1 + Z$$

de vecteur directeur $\overrightarrow{v_5}\binom{-1}{5}$. La droite qui a une ordonnée à l'origine maximale est celle qui contient le point $A\binom{7}{0}$. La fonction objectif atteint son maximum au point $(7,0)$ et vaut $Z(7,0) = 5\times 7 + 1\times 0 = 35$.



Remarque 1.1.17 *En fonction des différentes valeurs attribuées à c_1 et c_2, la fonction objectif atteint son maximum en différents sommets du polygone. Le programme linéaire a soit une unique solution soit une infinité de solutions (lorsque la droite D_Z est parallèle à l'un des côtés du polygone).*

Étude du cas général

L'équation de D_Z est donnée par :

$$D_Z : c_1x_1 + c_2x_2 = Z \Leftrightarrow x_2 = -\frac{c_1}{c_2}x_1 + \frac{Z}{c_2} \text{ avec } c_1 > 0, c_2 > 0.$$

Ces droites D_Z ont pour vecteur directeur $\vec{v}\binom{-c_2}{c_1}$, pour coefficient directeur $m = -\frac{c_1}{c_2}$ et pour ordonnée à l'origine $p = \frac{Z}{c_2}$.
Maximiser Z est équivalent à maximiser $\frac{Z}{c_2}$. On recherche une ou plusieurs droites D_Z rencontrant le domaine des solutions réalisables et ayant une ordonnée à l'origine maximale.

— Le côté (AB) a pour équation $2x_1 + x_2 = 14$, le coefficient directeur est -2 et $6 \leq x_1 \leq 7$.

— Le côté (BC) a pour équation $x_1 + x_2 = 8$, le coefficient directeur est -1 et $3 \leq x_1 \leq 6$.

— Le côté (CD) a pour équation $x_1 + 3x_2 = 18$, le coefficient directeur est $-\frac{1}{3}$ et $0 \leq x_1 \leq 3$.

La droite D_Z a pour coefficient directeur $-\frac{c_1}{c_2}$, on compare ensuite ce coefficient aux pentes des droites contenant les côtés $(AB), (BC)$ et (CD).

- $-\frac{c_1}{c_2} < -2 \Leftrightarrow \frac{c_1}{c_2} > 2 \Leftrightarrow c_1 > 2c_2$

Dans ce cas, la droite des bénéfices est plus "pointue" que le côté (AB). Le maximum est atteint au point $A(7,0)$ et en ce point seulement. Le programme linéaire admet une seule solution maximale $(7,0)$ qui est un sommet, avec $x_2 = 0$ on ne produit que P_1.

- $-\frac{c_1}{c_2} = -2 \Leftrightarrow c_1 = 2c_2$
  -2 est la pente du côté (AB). Les droites $D_Z : c_1x_1 + c_2x_2 = Z$ sont parallèles au côté (AB). Il y a une infinité de solutions optimales représentées par tous les points du segment $[AB]$ défini par :

$$[AB] : \begin{cases} 2x_1 + x_2 = 14 \\ 6 \leq x_1 \leq 7 \end{cases}$$

Tous les couples (x_1, x_2) tels que $\begin{cases} 6 \leq x_1 \leq 7 \\ 2x_1 + x_2 = 14 \end{cases}$ sont solutions optimales, le bénéfice vaut alors $14c_2$. En effet, $Z(x_1, x_2) = c_1x_1 + c_2x_2 = 2c_2x_1 + c_2x_2 = c_2(2x_1 + x_2)$.

- $-2 < -\frac{c_1}{c_2} < -1 \Leftrightarrow 1 < \frac{c_1}{c_2} < 2$
  -1 est la pente du côté (BC), -2 celle de (AB). Le maximum est atteint en un seul point B qui est aussi un sommet.
- $-\frac{c_1}{c_2} = -1 \Leftrightarrow \frac{c_1}{c_2} = 1 \Leftrightarrow c_1 = c_2$

Les droites D_Z sont parallèles au côté (BC). Il y a une infinité de solutions optimales représentées par tous les points du segment $[BC]$ défini par :

$$[BC] : \begin{cases} x_1 + x_2 = 8 \\ 3 \leq x_1 \leq 6 \end{cases}$$

Tous les couples (x_1, x_2) tels que $\begin{cases} 3 \leq x_1 \leq 6 \\ x_1 + x_2 = 8 \end{cases}$ sont solutions optimales, le bénéfice vaut alors $8c_1$.

- $-1 < -\frac{c_1}{c_2} < -\frac{1}{3}$
  $-\frac{1}{3}$ est la pente du côté (CD), -1 celle du côté (BC). Le programme linéaire a un seule solution optimale soit le point $C(3,5)$ qui est un sommet.
- $-\frac{c_1}{c_2} = -\frac{1}{3} \Leftrightarrow c_2 = 3c_1$

Les solutions optimales sont tous les points du segment $[CD]$ d'où une infinité de solutions.

$$[CD] : \begin{cases} x_1 + 3x_2 = 18 \\ 0 \leq x_1 \leq 3 \end{cases}$$

La fonction objectif atteint son maximum pour tous les couples (x_1, x_2) tels que

$$\begin{cases} x_1 + 3x_2 = 18 \\ 0 \leq x_1 \leq 3 \end{cases}$$

et le bénéfice vaut $Z = 18c_1$.

- $-\frac{1}{3} < -\frac{c_1}{c_2} < 0 \Leftrightarrow 0 < \frac{c_1}{c_2} < \frac{1}{3}$.

Il existe une seule solution optimale c'est-à-dire le point $D(0,6)$ qui est un sommet ; x_1 étant nul, on ne produit que P_2.

Exemple 1.1.18 *Considérons l'exemple suivant faisant intervenir trois dimensions :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \\ 2x_1 + x_2 + 2x_3 \leq 4 \\ Z(x_1, x_2, x_3) = x_1 + x_2 \ \grave{a} \ maximiser \end{cases} \tag{1.12}$$

On trace le plan d'équation $2x_1 + x_2 + 2x_3 = 4$. Ce plan rencontre les axes de coordonnées aux points $M_1 \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}, M_2 \begin{pmatrix} 0 \\ 4 \\ 0 \end{pmatrix}, M_3 \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}$ (voir Figure 1.12).

Le domaine des solutions réalisables est représenté par l'intérieur de la pyramide $OM_1M_2M_3$. La

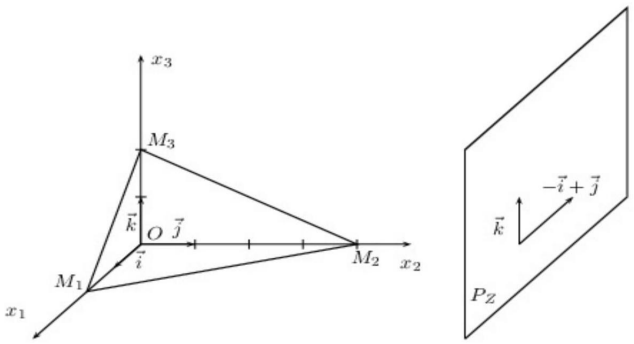


FIGURE 1.12 – Domaine des solutions réalisables du système (1.12)

fonction objectif est $Z(x_1, x_2, x_3) = x_1 + x_2$. Lorsque Z varie, $x_1 + x_2 = Z$ est l'équation d'un plan parallèle à $(0, \vec{k})$, ce plan rencontre le plan $(O, \vec{i}, \vec{j})$ suivant la droite d'équation $Z = 0$ et $x_1 + x_2 = Z$. Le plan P_Z qui rencontre le domaine des solutions réalisables et tel que Z soit maximum est celui qui contient le point $M \begin{pmatrix} 0 \\ 4 \\ 0 \end{pmatrix}$. La fonction objectif atteint son maximum en un seul point qui est d'ailleurs un des sommets, c'est-à-dire M_2.

1.1.5.5 Cas général

Soit un programme linéaire $\mathcal{P}$. On admettra les résultats suivants :

1. Le domaine des solutions réalisables de tout programme linéaire à n variables est soit l'ensemble vide $\emptyset$ soit une partie convexe de $\mathbb{R}^n$.
2. Dans le cas d'un programme linéaire à deux variables, le domaine des solutions réalisables, lorsqu'il n'est pas vide, est une partie D du plan délimité par un polygone convexe, possédant éventuellement des côtés de longueur infinie.
   Dans chaque cas, l'ensemble des solutions optimales (lorsqu'il n'est pas vide) contient un sommet de D, c'est-à-dire que si la fonction objectif a un maximum ou un minimum, il est atteint en au moins un des sommets du polygone délimitant le domaine des solutions réalisables.
3. On admettra que ces résultats se généralisent à un programme linéaire à n variables.

1.1.6 Exercices

Exercice 1. Formaliser les situations suivantes :

(a) La société Bonvin, S.A., qui pratique le négoce en vins propose à sa clientèle deux vins de table : l'un est dénommé "Extra", l'autre "Supérieur". Ces produits sont obtenus par coupage de crus issus de diverses régions : un vin de l'Hérault, un vin du Bordelais et un vin d'Italie.
Les coupages sont réalisés selon les proportions fixes suivantes :

	Vin "Extra"	Vin "Supérieur"
Vin de l'Hérault	0,5	0,2
Vin du Bordelais	0,3	0,6
Vin d'Italie	0,2	0,2
Total	1	1

Après les vendanges, la société dispose en stock dans ses cuves des quantités suivantes de crus d'origine :

Vin de l'Hérault	..	13600 hectolitres
Vin du Bordelais	..	12000 hectolitres
Vin d'Italie	...	10400 hectolitres

Ces quantités constituent les ressources disponibles pour la production de l'année à venir. En outre, compte tenu des capacités techniques de mise en bouteille existantes, cette production ne peut pas dépasser 36000 hectolitres au total dans l'année.
L'activité de cette entreprise comporte des coûts qui ont été classés en deux catégories :

— Une partie est considérée comme fixe ; elle correspond aux approvisionnements, puisque ceux-ci sont déja constitués, ainsi qu'aux frais de personnel. Ces coûts s'élèvent à 12000000 euros pour l'année.

— L'autre partie correspond aux frais de mise en bouteille, d'emballage et de commercialisation. Cette seconde partie est proportionnelle aux quantités produites : soit 100 euros par hectolitre de vin quelle que soit la qualité de celui-ci.
Une étude de marché révèle que celui-ci ne saurait absorber plus de

— 20000 hectolitres de vin "Extra" à 500 euros par hectolitre,

— et 16000 hectolitres de vin "Supérieur" à 600 euros l'hectolitre.

Le problème de cette entreprise peut être formulé ainsi :
Quelles quantités faut-il produire de vin "Extra" et "Supérieur" afin de rendre maximum le bénéfice total ?

(b) Considérons désormais :

— que le vin "Extra" doit contenir au moins 30% de cru du Bordelais et au plus 20% de cru d'Italie,

— et que le vin "Supérieur" doit être composé d'au moins 60% de cru du Bordelais et d'au moins 20% de cru de l'Hérault.
Toutes les autres caractéristiques du problème restent identiques au cas précédent.
Le problème peut s'exprimer sous la forme : *Quelle quantité de chaque vin d'origine affecter à chaque qualité de produit fini ?*

(c) On considère un coût d'approvisionnement qui n'est plus fixe. Transport inclus, il s'élève

à :

vin de l'Hérault : 230 euros l'hectolitre,
vin du Bordelais : 250 euros l'hectolitre,
vin d'Italie : 180 euros l'hectolitre.

Il subsiste néanmoins un coût fixe constitué pour l'essentiel de frais de personnel, égal à 4000000 euros. Le problème présent comporte trois questions :

- Quelle quantité produire
- Quelle composition adopter

} pour chaque vin, "Extra" et "Supérieur",

- Quelle quantité de matières premières acquérir auprès des fournisseurs ?

Ces trois questions sont liées et on peut constater que le fait de connaître la quantité de chaque matière première incorporée dans chaque produit permet de déterminer simultanément l'approvisionnement nécessaire, la composition adéquate des produits et la quantité à produire.

(d) Les produits de la société sont conditionnés dans des récipients de 0,75 litre et de 3 litres. Afin de pouvoir satisfaire la clientèle, Bonvin se fixe comme objectif annuel de disposer d'au moins 400000 bouteilles de 3 litres et d'au moins 3200000 bouteilles de 0,75 litre. Pour produire ces récipients Bonvin dispose de deux ateliers dont les rendements sont différents :

Nombre de récipients par heure de fonctionnement

	Atelier A	Atelier B
0,75 litre ...	500	400
3 litres	400	320

Chaque atelier fonctionne au maximum 4000 heures dans l'année. Les prévisions de coût variable de production de chaque type de récipient donnent comme résultats :

Coûts variables de production

	Atelier A	Atelier B
0,75 litre ...	0,4	0,55
3 litres	0,75	0,85

Mais *Bonvin* peut également sous-traiter la fabrication de ces récipients à la société Corec qui propose comme tarif :

$0,5$ euro la bouteille de $0,75$ litre
1 euro la bouteille de 3 litres

Les dirigeants de Bonvin S.A. se posent trois questions

— faut-il produire des bouteilles et en quelles quantités ?
— en utilisant quelle technique de production (atelier A et/ou atelier B) ?
— faut-il sous-traiter tout ou partie de la production à Corec ?
qui peuvent être condensées en une seule :

Quelles filières utiliser pour obtenir les bouteilles nécessaires ?

Exercice 2. Une entreprise stocke successivement deux types de polystyrènes A_1 et A_2 dans trois entrepôts distincts E_1, E_2 et E_3 afin qu'ils y subissent des traitements particuliers. Le coût de fonctionnement de l'entrepôt E_1 est de 200 euros par jour, celui de l'entrepôt E_2 est de 400 euros et celui de l'entrepôt E_3 est de 300 euros. Les temps de stockage pour une tonne de polystyrène A_1 sont de 3 jours dans l'entrepôt E_1, de 1 jour dans l'entrepôt E_2 et d'une demi-journée dans l'entrepôt E_3. Ils sont pour le polystyrène A_2 de 2 jours dans chacun des 3 entrepôts.
Les coûts de fabrication des polystyrènes A_1 et A_2 sont respectivement de 600 euros et 400 euros la tonne. Les prix de vente d'une tonne des polystyrènes fabriqués sont de 1950 euros pour A_1 et de 2440 euros pour A_2.

(1) (a) . Calculer le coût de stockage d'une tonne de polystyrène A_1 et d'une tonne de polystyrène A_2.

item[(b)]. Déterminer le bénéfice réalisé par la fabrication, le stockage et la vente d'une tonne de chacun des produits.

(c) . En déduire que le bénéfice total Z pour la production, le stockage et la vente de x tonnes de polystyrène A_1 et de y tonnes de polysytyrène A_2 est donné par $Z(x, y) = 200x + 240y$.

(2) La logistique des stockages est telle que l'entrepôt E_1 peut fonctionner au maximum 360 jours dans l'année, l'entrepôt E_2 peut fonctionner au maximum 160 jours par an, l'entrepôt E_3 ne peut fonctionner annuellement plus de 120 jours.
La demande est telle que la production de polystyrène A_1 ne peut dépasser 120 tonnes, celle de A_250 tonnes.

(a) Déterminer les nombres x et y de tonnes des deux produits fabriqués pour que l'entrepôt E_1 fonctionne exactement 360 jours et l'entrepôt E_3 exactement 120 jours. Cette production est-elle possible ?

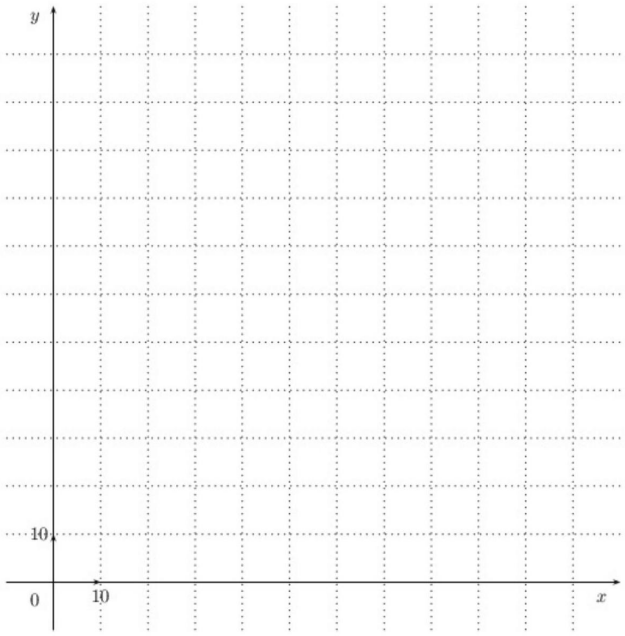
(b) On veut maintenant déterminer les nombres x et y de tonnes des deux produits fabriqués, stockés et vendus qui donneraient à l'entreprise le bénéfice maximum.

(i) Donner les 7 contraintes de production ainsi que la fonction à maximiser sous la forme d'un programme linéaire du type

$$\begin{cases} x \leq \cdots \text{ et / ou } \geq \cdots \\ y \leq \cdots \text{ et / ou } \geq \cdots \\ \vdots \\ Z(x, y) = \cdots \text{ à maximiser} \end{cases}$$

(ii) . Représenter sur le graphique ci-joint le domaine des solutions réalisables en justifiant.

(iii) . À l'aide d'une résolution graphique, déterminer en justifiant la production qui assurera le bénéfice maximal. Quel sera alors son prix ?



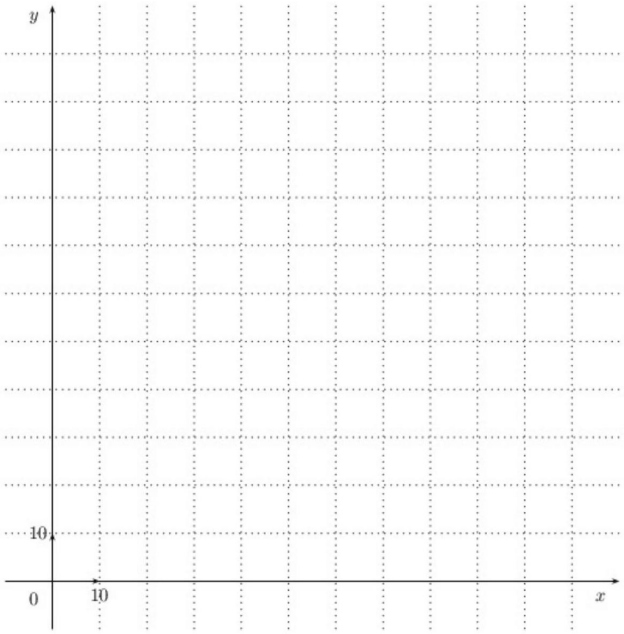
Exercice 3. Une entreprise possède deux unités de production U_1 et U_2. Elle commercialise ses produits à l'aide de trois entrepôts distincts E_1, E_2 et E_3 situés dans différentes zones de consommation. Le tableau cidessous indique pour chaque entrepôt, les proportions de stockage d'unités x et y provenant respectivement de U_1 et U_2.

	E_1	E_2	E_3
U_1	1	2	1
U_2	2	3	1

Ces valeurs signifient par exemple que les structures de l'entrepôt E_1 permettent de stocker 2 fois plus d'unités provenant de U_2 que d'unités provenant de U_1. L'organisation actuelle des entrepôts est telle que E_1 ne peut stocker au total plus de 120 unités, E_2 ne peut stocker au total plus 200 unités et E_3 ne peut stocker au total plus 90 unités.
Les productions journalières de U_1 et de U_2 sont limitées respectivement à 80 et 50 unités.
On sait que le bénéfice réalisé par l'entreprise est de 50 euros pour la vente d'une unité de U_1 et 80 euros pour la vente d'une unité de U_2.
On veut déterminer maintenant les nombres x et y d'unités provenant de U_1 et U_2, qui permettraient à l'entreprise de réaliser un bénéfice journalier maximum.

(a) Donner les 7 contraintes portant sur x et y ainsi que la fonction à maximiser sous la forme d'un programme linéaire

(b) Résolution graphique
(i) Représenter sur le graphique de la page suivante, le domaine des solutions réalisables en justifiant vos démarches.
(ii) À l'aide d'une résolution graphique, déterminer en justifiant, la production qui assurera le bénéfice maximal. À quoi sera alors égal ce bénéfice ?



Exercice 4. Le gérant d'un entrepôt souhaite renouveler le matériel de sécurité de son établissement. Il a besoin au minimum de

- 90 paires de chaussures de sécurité,
- 40 casques de sécurité,
- 240 paires de gants.

Une première entreprise de vente lui propose un lot A comprenant 2 paires de chaussures, 4 casques et 8 paires de gants pour 200 euros. Une deuxième entreprise vend pour 400 euros un lot B de 3 paires de chaussures, 12 casques et 6 paires de gants.
Pour répondre à ses besoins, le gérant achète x lots A et y lots B .

(1) Traduire par un système d'inéquations les contraintes auxquelles satisfont x et y.
On considère un plan P rapporté à un repère orthonormé $(O, \vec{i}, \vec{j})$. A tout couple (x, y) on associe le point M de P de coordonnées (x, y), en prenant comme unité 1 cm pour 10 lots.

(2) Représenter dans P l'ensemble des points $M(x, y)$ satisfaisant aux inéquations :

$$\begin{cases} x \geq 0 \text{ et } y \geq 0 \\ 2x + 3y \geq 90 \\ x + 3y \geq 60 \\ 4x + 3y \geq 120 \end{cases}$$

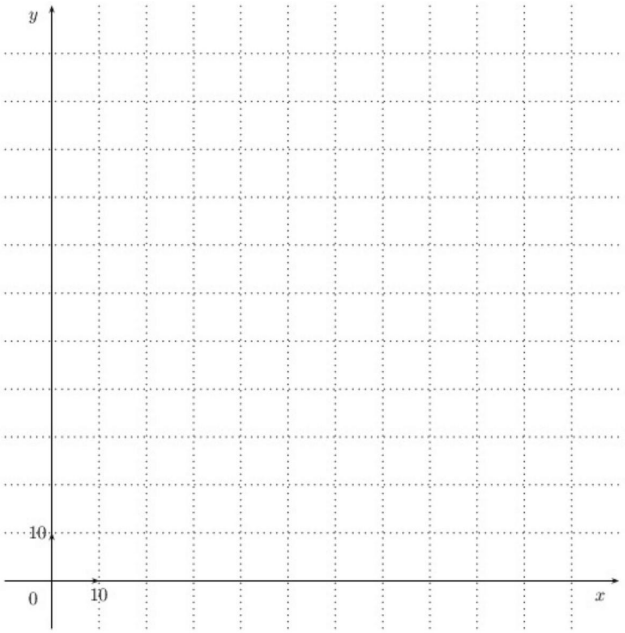
On hachurera la partie du plan formée des points pour lesquels les contraintes ne sont pas respectées.

(3) Exprimer en fonction de x et de y la dépense en euros occasionnée par l'achat de x lots A et de y lots B.

(4) Est-il possible de procéder aux achats nécessaires avec 5000 euros ? Justifier la réponse.

(5) Déterminer graphiquement, en précisant la démarche suivie, le nombres de lots A et de lots B à acheter pour avoir une dépense minimale.

(6) Quelle est cette dépense minimale ?



Exercice 5. Un artisan fabrique des objets A et des objets B. On dispose des informations suivantes :

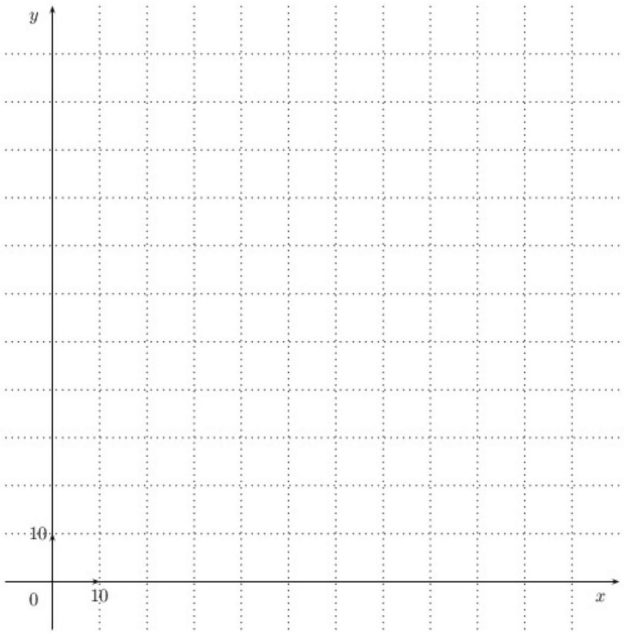
- La réalisation d'un objet A demande 30 euros de matière première et 125 euros de main-d'uvre.
- La réalisation des objets B demande 70 euros de matière première et 75 euros de mains-d'uvre.
- Les profits réalisés sont de 54 euros par objets A, et de 45 euros par objet B .

On note x le nombre d'objets A fabriqués et y le nombre d'objets B fabriqués, en une journée. La dépense journalière en matière première ne doit pas dépasser 560 euros. La dépense journalière en main-d'uvre ne doit pas dépasser 1250 euros.

(1) Traduire mathématiquement ces deux hypothèses.

(2) Le plan est rapporté à un repère orthonormé (unité graphique = 1 cm). Représenter graphiquement l'ensemble des points $M(x, y)$ dont les coordonnées vérifient ces hypothèses. Exprimer le bénéfice
journalier Z de l'entreprise en fonction de x et de y, puis la production journalière d'objets A et B qui assurerait un bénéfice maximum. On précisera, graphiquement, et par le calcul, cette production journalière.

(3) En déduire le montant de ce bénéfice.



Exercice 6. Résoudre le problème de la société Bonvin S.A. dans sa forme initiale à l'aide de la méthode graphique.

Exercice 7. Nous prenons un exemple tiré de Hillier et Lieberman. Il s'agit d'une entreprise de fabrication de chassis qui envisage la production de deux nouveaux modèles au moyen des capacités résiduelles de ses trois ateliers. Il s'agit respectivement d'un chassis en aluminium et d'un chassis en bois. Le premier produit nécessite le passage dans le premier atelier pour fabriquer le cadre en aluminium et dans le troisième atelier où le verre est monté sur le chassis. Tandis que le second produit nécessite le passage dans le deuxième atelier pour fabriquer le cadre en bois et dans le troisième atelier où le verre est monté sur le chassis. Les marges unitaires, les temps de fabrication de chacun des produits dans chacun des ateliers ainsi que les capacités hebdomadaires résiduelles de ces ateliers sont donnés au tableau ci-dessous. Combien faut-il produire

	Produit 1 (heures/produit)	Produit 2 (heures/produit)	Capacité disponible (heures/semaine)
Atelier 1	1	0	4
Atelier 2	0	2	12
Atelier 3	3	2	18
Marge	3$	5$	

de chassis de chaque type par semaine pour maximiser le profit net ?

Exercice 8. Une société de tri de déchets et recyclage de papier peut se fournir en déchets auprès de deux villes. Son rôle consiste à séparer les listes d'ordinateur et les journaux. La répartition entre ménages et sociétés est différente d'une ville à l'autre expliquant un pourcentage différent de listes d'ordinateur et de journaux dans les déchets. Ces pourcentages ainsi que la quantité maximum de déchets que peuvent fournir par an ces deux villes sont reprises au tableau suivant : La société offre aux villes un prix de 35€ par tonne

	Listes (%)	Journaux (%)	Offre (tonnes par an)
Ville 1	5	20	10000
Ville 2	15	30	20000

de déchet. Elle doit décider du montant optimal de déchets à acheter à chaque ville pour minimiser son coût d'achat. Pour couvrir ses frais fixes, la société doit au moins collecter 1500 tonnes de listing d'ordinateur par an. La société ne desire pas collecter plus de 6000 tonnes de journaux par an. Combien la société doit-elle acheter de déchets par an à chacune des villes ?

1. Formuler mathématiquement le problème (choix des variables, expression des contraintes et de l'objectif).
2. Déterminer graphiquement le plan d'achat optimal et en déduire le coût d'achat minimum.

Exercice 9. Une entreprise fabrique deux produits P_1 et P_2. Chaque produit doit passer les deux ateliers d'usinage et de finition. Le mois dernier, 500 unités de P_1 ont été produites grâce à 750 heures d'usinage et 250 heures de finition. De même, 700 unités de P_2 ont été produites, nécessitant 700 heures d'usinage et 350 heures de finition. Une partie du coût de production est indépendante du nombre d'heures passées à la production (les frais fixes), une partie est directement proportionnelle au nombre d'heures passées à la production (les frais variables). Le mois passé, on a observé la répartition suivante entre frais fixes et frais variables : Il y a un coût de conditionnement de 8€ l'unité pour P_1 et de 6€ pour P_2. Les prix de vente sont

Section	Frais fixes	Frais variables
Usinage	60000	11600
Finition	40000	6000

de 55€ et 43€ respectivement.

1. Calculer les marges sur coûts variables (différence entre prix de vente et coût variable de production) par unité de chacun des deux produits. Indication : calculer d'abord le prix de l'heure dans chacun des ateliers et le temps nécessaire dans chacun des ateliers par produit.
2. Les capacités de production sont de 1200 heures par mois pour l'usinage et de 500 heures pour la finition. Formuler le programme linéaire correspondant à la maximisation de la marge sur coûts variables.
3. Déterminer graphiquement la solution optimale.

1.2 La programmation linéaire - Méthode du simplexe

1.2.1 Introduction

L'algorithme du simplexe fut proposé en 1947 par G. B. Dantzig comme méthode de résolution générale des programmes linéaires. La solution optimale est approchée par étapes ou itérations successives. Chaque étape correspond au calcul de la valeur économique d'une solution. Comme il existe une infinité de solutions admissibles, la méthode propose de n'explorer qu'un nombre limité de solutions parmi lesquelles se trouve à coup sûr la solution optimale.

1.2.2 La méthode du simplexe

La méthode du simplexe repose sur le théorème fondamental suivant :

Théorème 1.2.1 — *Si un programme linéaire admet une solution possible finie, alors il admet au moins une solution de base.*

— *Si ce programme linéaire admet une solution optimale, il admet au moins une solution de base optimale (ce qui signifie qu'une solution de base au moins est optimale).*

La solution optimale étant une solution de base, l'algorithme du simplexe consiste à :

1. déterminer une solution de base,
2. faire subir un test d'optimalité à cette solution de base pour déterminer s'il s'agit ou non de la solution optimale,

— s'il s'agit de la solution optimale, le problème est terminé,

— s'il ne s'agit pas de la solution optimale, on passe à l'étape 3.,

3. changer de solution de base puis reprendre la procédure au 1. jusqu'à l'obtention de la solution optimale. Chaque changement de solution de base constitue une itération.
   Afin de réaliser les opérations successives de l'algorithme du simplexe, il convient de mettre le programme sous une forme standard.

1.2.3 Programme linéaire standard

Exemple 1.2.2 *On se donne le problème suivant :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ 5x_1 - x_2 \leq 3 \\ x_1 + 4x_2 \leq 4 \\ Z(x_1, x_2) = 2x_1 + 3x_2 \ \textit{à optimiser}, \end{cases} \tag{1.13}$$

programme linéaire exprimé sous sa forme canonique.

On introduit des variables auxiliaires positives ou nulles appelées **variables d'écart** de la façon suivante :

$$5x_1 - x_2 \leq 3 \Leftrightarrow \begin{cases} 5x_1 - x_2 + e_1 = 3 \\ e_1 \geq 0 \end{cases} \quad \text{et} \quad x_1 + 4x_2 \leq 4 \Leftrightarrow \begin{cases} x_1 + 4x_2 + e_2 = 4 \\ e_2 \geq 0 \end{cases}$$

Le programme linéaire peut se réécrire alors :

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, e_1 \geq 0, e_2 \geq 0 \\ 5x_1 - x_2 + e_1 = 3 \\ x_1 + 4x_2 + e_2 = 4 \\ Z(x_1, x_2) = 2x_1 + 3x_2 \text{ à optimiser.} \end{cases}$$

Le programme est écrit sous sa **forme standard** et les variables e_1 et e_2 sont des variables d'écart.

Exemple 1.2.3 *On se donne le programme linéaire ci-dessous :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \\ x_1 + x_2 \leq 1 \\ x_1 + 2x_2 + 3x_3 \leq 5 \\ x_2 - 4x_3 \leq 2 \\ x_1 + x_2 + x_3 = 5 \\ Z(x_1, x_2, x_3) = 2x_1 + x_2 + x_3 \text{ à optimiser.} \end{cases} \tag{1.14}$$

On remplace les 3 inégalités par 3 égalités en introduisant 3 variables d'écart e_1, e_2 et e_3. Le programme linéaire standard est alors

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, e_1 \geq 0, e_2 \geq 0, e_3 \geq 0 \\ x_1 + x_2 + e_1 = 1 \\ x_1 + 2x_2 + 3x_3 + e_2 = 5 \\ x_2 - 4x_3 + e_3 = 2 \\ x_1 + x_2 + x_3 = 5 \\ Z(x_1, x_2, x_3) = 2x_1 + x_2 + x_3 \text{ à optimiser.} \end{cases}$$

Cas général
Soit un programme linéaire à n variables. On remplace chaque inégalité

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b_1$$

par l'égalité

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n + e_1 = b_1 \text{ avec } e_1 \geq 0$$

et

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b_1$$

par

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n - e_1 = b_1 \text{ avec } e_1 \geq 0$$

On obtient alors le programme linéaire standard qu'on cherche à résoudre.

1.2.3.1 L'algorithme du simplexe

Exemple 1.2.4 *(solution unique)*

1. *Enoncé*
   Un ébéniste fabrique des bureaux sous forme standard ou luxe. Des études de marché ont montré que pour l'année à venir, les possibilités de vente s'élèvent à 300 unités pour le modèle luxe et à 400 unités pour le modèle standard. L'approvisionnement en bois est suffisant pour fabriquer annuellement 500 bureaux quel que soit le type. Par ailleurs, le temps de fabrication d'un modèle luxe est le double de celui d'un bureau de modèle standard. La capacité annuelle de fabrication est telle que, si tous les bureaux fabriqués étaient de type standard, on pourrait en fabriquer 700 au maximum. La vente d'un bureau sous le modèle luxe conduit à une marge unitaire sur coût variable égale à 7, celle d'un bureau de type standard égale à 5 . On se propose de rechercher le programme annuel de fabrication conduisant au profit global maximum.

2. *Mise en équation*
   Soit x_1 le nombre de bureaux de type luxe, x_2 le nombre de bureaux de type standard. Le programme linéaire est

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 \leq 300 \\ x_2 \leq 400 \\ x_1 + x_2 \leq 500 \\ 2x_1 + x_2 \leq 700 \\ Z(x_1, x_2) = 7x_1 + 5x_2 \text{ à maximiser} \end{cases} \tag{1.15}$$

3. *Domaine des solutions réalisables*

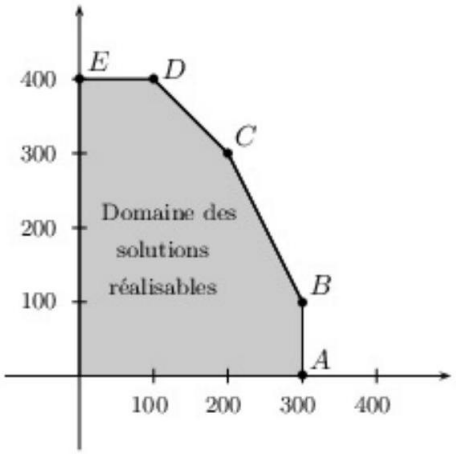


FIGURE 1.13 – Domaine des solutions réalisables du système (1.15).

4. *Forme standard*
   On introduit les variables d'écart x_i avec $i \in \{3, 4, 5, 6\}$ positives ou nulles.

$$\begin{cases} x_1 + x_3 = 300 \\ x_2 + x_4 = 400 \\ x_1 + x_2 + x_5 = 500 \\ 2x_1 + x_2 + x_6 = 700 \\ Z(x_1, x_2) = 7x_1 + 5x_2 \text{ à maximiser} \end{cases}$$

5. *Variables hors-base, variable dans la base*
   Une solution de base est avant tout une solution admissible ; elle satisfait l'ensemble des contraintes et conditions de signe. Toute solution de base comporte deux catégories de variables.
   - Des variables ayant une valeur prédéterminée nulle : ces variables nulles sont dites variables horsbase (ou variables exclues). Il y a au moins autant de variables hors-base que le problème comporte de variables réelles.
   - Des variables ayant une valeur non nulle : ce sont les variables dans la base (ou variables retenues). Leur nombre est au plus équivalent au nombre de variables d'écart.
     De façon générale, si un problème comprend m contraintes et n variables réelles, pour qu'une solution soit solution de base il faut et il suffit

— qu'elle soit solution admissible,
— qu'elle admette au moins n variables hors base et au plus m variables dans la base.

Pour amorcer l'algorithme du simplexe, il est nécessaire de connaître une solution de base.
La solution de base de départ de l'ébéniste consiste à ne rien produire : $x_1 = x_2 = 0$. Ces variables x_1, x_2 qui sont nulles sont hors-base. Dans ce cas, $x_3 = 300, x_4 = 400, x_5 = 500, x_6 = 700$. Les variables x_3, x_4, x_5, x_6 non nulles sont dans la base. La valeur de la fonction économique est $Z(0,0) = 7 \times 0 + 5 \times 0 = 0$.
Notation :

VDB	VHB
x_3	x_1
x_4	x_2
x_5	
x_6	

Tableau initial :

VHB / VDB	x_1	x_2	x_3 •	x_4 •	x_5 •	x_6 •	cste	
x_3	1	0	1	0	0	0	300	$x_1 + x_3 = 300$
x_4	0	1	0	1	0	0	400	$x_2 + x_4 = 400$
x_5	1	1	0	0	1	0	500	$x_1 + x_2 + x_5 = 500$
x_6	2	1	0	0	0	1	700	$2x_1 + x_2 + x_6 = 700$
Z	7	5	0	0	0	0	0	$Z = 7x_1 + 5x_2$

6. *Première itération*
La solution de base de départ consiste à ne rien produire soit $x_1 = x_2 = 0$. On étudie ensuite, à partir de cette solution, jusqu'à quel niveau on peut porter x_1 ou x_2 conformément aux contraintes de façon à accroître au maximum le profit. Il se pose le problème du choix de la variable x_1 ou x_2 qui va passer de la valeur 0 à une valeur strictement positive. La variable choisie sera appelée variable entrante.

- Critère de sélection de la variable entrante :

Cette sélection doit s'accompagner d'une augmentation de la fonction économique

$$Z(x_1, x_2) = 7x_1 + 5x_2$$

La sélection portera sur x_1 qui par unité rapporte le plus. Cette règle est appelée **règle du plus grand gain marginal** :

> Le critère de sélection de Dantzig de la variable entrante consiste, dans la fonction économique exprimée exclusivement en fonction des variables hors-base, à sélectionner la variable affectée du coefficient strictement positif le plus élevé.

- On exprime ensuite x_3, x_4, x_5, x_6 et Z en fonction des variables hors-base x_1 et x_2

$$\begin{cases} x_3 = 300 - x_1 \\ x_4 = 400 - x_2 \\ x_5 = 500 - x_1 - x_2 \\ x_6 = 700 - 2x_1 - x_2 \\ Z = 7x_1 + 5x_2 \end{cases}$$

La variable x_2 reste hors-base donc nulle, la variable x_1 entre en base. On reporte $x_2 = 0$ dans ce système, on obtient :

$$\begin{cases} x_3 = 300 - x_1 \\ x_4 = 400 \\ x_5 = 500 - x_1 \\ x_6 = 700 - 2x_1 \\ Z = 7x_1 \end{cases}$$

On cherche jusqu'à quel niveau il est possible de porter x_1, de façon compatible avec les contraintes $x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0$. Les contraintes de positivité donnent

$$x_1 \leq 300, x_1 \leq 500, x_1 \leq 350.$$

La valeur maximale prise par x_1 est donc 300 . On remplace x_1 par 300 dans le système et on obtient

$$x_3 = 0, x_4 = 400, x_5 = 200, x_6 = 100 \text{ et } Z(300, 0) = 2100.$$

La variable x_3 est devenue nulle, elle est sortie de la base, x_3 est appelée variable sortante. Les variables x_1 et x_3 ont permuté.

VDB	VHB
x_1	x_3
x_4	x_2
x_5	
x_6	

On exprime le programme standard en fonction des nouvelles variables hors-base x_2, x_3 :

$$\begin{cases} x_1 + x_3 = 300 \\ x_2 + x_4 = 400 \\ x_1 + x_2 + x_5 = 500 \\ 2x_1 + x_2 + x_6 = 700 \\ Z = 7x_1 + 5x_2 \end{cases} \Leftrightarrow \begin{cases} x_1 = 300 - x_3 \\ x_4 = 400 - x_2 \\ x_5 = 500 - (300 - x_3) - x_2 \\ x_6 = 700 - 2(300 - x_3) - x_2 \\ Z = 7(300 - x_3) + 5x_2 \end{cases} \Leftrightarrow \begin{cases} x_1 + x_3 = 300 \\ x_2 + x_4 = 400 \\ x_2 - x_3 + x_5 = 200 \\ x_2 - 2x_3 + x_6 = 100 \\ Z = 5x_2 - 7x_3 + 2100 \end{cases}$$

On exprime ce nouveau programme à l'aide d'un second tableau. Pour l'obtenir, on remplace impérativement dans le premier tableau la variable x_3 par la variable x_1 (x_1 et x_3 ont permuté) et ceci dans la colonne "variables dans la base".

VHB / VDB	x_1	x_2	x_3	x_4 •	x_5 •	x_6 •	cste	
x_1	1	0	1	0	0	0	300	$x_1 + x_3 = 300$
x_4	0	1	0	1	0	0	400	$x_2 + x_4 = 400$
x_5	0	1	-1	0	1	0	200	$x_2 - x_3 + x_5 = 200$
x_6	0	1	-2	0	0	1	100	$x_2 - 2x_3 + x_6 = 100$
Z	0	5	-7	0	0	0	-2100	$Z = 5x_2 - 7x_3 + 2100$

On a pris la colonne des variables dans la base du premier tableau et on y a remplacé x_3 par x_1.

Pour la fonction économique Z, le coefficient constant 2100 est affecté impérativement du signe "-" et on place -2100.

7. *Deuxième itération*

   - Sélection de la variable entrante :

$$Z = 5x_2 - 7x_3 + 2100$$

On sélectionne x_2 ; en effet, toute augmentation de x_3 à partir de la valeur 0 provoquerait une diminution de la fonction économique Z.

- Sélection de la variable sortante : la variable x_3 reste hors-base donc nulle, on remplace x_3 par 0 dans le système précédent, on obtient $x_1 = 300, x_4 = 400 - x_2 \geq 0, x_5 = 200 - x_2 \geq 0$ et $x_6 = 100 - x_2 \geq 0$. Les contraintes de positivité imposent

$$x_2 \leq 400, x_2 \leq 200 \text{ et } x_2 \leq 100.$$

Jusqu'à quel niveau peut-on porter x_2 ? La valeur maximale prise par x_2 est 100 . Dans ce cas,

$$x_1 = 300, x_4 = 300, x_5 = 100 \text{ et } x_6 = 0.$$

La variable sortante est x_6.
Les variables hors-base sont alors x_3 et x_6, les variables dans la base sont x_1, x_2, x_4 et x_5. Cette itération conduit au sommet $B(300, 100)$. Pour cette solution, la fonction économique prend la valeur 2600 .

VDB	VHB
x_1	x_3
x_2	x_6
x_4	
x_5	

x_2 et x_6 ont permuté. On exprime les variables dans la base en fonction des nouvelles variables hors-base x_3 et x_6

$$\begin{cases} x_1 + x_3 = 300 \\ x_2 + x_4 = 400 \\ x_1 + x_2 + x_5 = 500 \\ 2x_1 + x_2 + x_6 = 700 \\ Z = 7x_1 + 5x_2 \end{cases} \Leftrightarrow \begin{cases} x_1 = 300 - x_3 \\ x_2 = 700 - 2(300 - x_3) - x_6 = 100 + 2x_3 - x_6 \\ x_4 = 400 - (100 + 2x_3 - x_6) \\ x_5 = 500 - (300 - x_3) - (100 + 2x_3 - x_6) \\ Z = 7(300 - x_3) + 5(100 + 2x_3 - x_6) \end{cases}$$

Le programme linéaire se réécrit finalement :

$$\begin{cases} x_1 + x_3 = 300 \\ x_2 - 2x_3 + x_6 = 100 \\ 2x_3 + x_4 - x_6 = 300 \\ x_3 + x_5 - x_6 = 100 \\ Z = 2600 + 3x_3 - 5x_6 \end{cases}$$

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste	
x_1	1	0	1	0	0	0	300	$x_1 + x_3 = 300$
x_4	0	0	2	1	0	-1	300	$2x_3 + x_4 - x_6 = 300$
x_5	0	0	1	0	1	-1	100	$x_3 + x_5 - x_6 = 100$
x_2	0	1	-2	0	0	1	100	$x_2 - 2x_3 + x_6 = 100$
Z	0	0	3	0	0	-5	-2600	$Z = 3x_3 - 5x_6 + 2600$

On a pris la colonne des variables dans la base du second tableau et on y a remplacé x_6 par x_2 (ces deux variables permutent).
Pour la fonction économique Z, le coefficient constant 2600 est affecté du signe "-" et on place -2600.

8. *Troisième itération*
   - Sélection de la variable entrante :

$$Z = 3x_3 - 5x_6 + 2600$$

x_3 sera la variable entrante car toute augmentation de x_6 entraîne une diminution de la fonction économique Z.

   - Sélection de la variable sortante : on exprime les variables dans la base en fonction des variables hors-base x_3 et x_6.

$$\begin{cases} x_1 = 300 - x_3 \\ x_2 = 100 + 2x_3 - x_6 \\ x_4 = 300 - 2x_3 + x_6 \\ x_5 = 100 - x_3 + x_6 \end{cases}$$

La variable x_6 reste hors-base donc nulle, on remplace x_6 par 0 . On obtient $x_1 = 300 - x_3 \geq 0$, $x_2 = 100 + 2x_3 \geq 0, x_4 = 300 - 2x_3 \geq 0$ et $x_5 = 100 - x_3 \geq 0$. Les contraintes de positivité donnent $x_3 \leq 300, x_3 \geq -50, x_3 \leq 150$ et $x_3 \leq 100$.
La valeur maximale prise par x_3 est 100 . Pour $x_3 = 100$, on obtient

$$x_1 = 200, x_2 = 300, x_4 = 100 \text{ et } x_5 = 0.$$

La variable qui sort de la base est x_5.
Cette itération conduit au sommet $C(200, 300)$. La valeur de la fonction économique est $Z = 2900$.

VDB	VHB
x_3	x_5
x_1	x_6
x_2	
x_4	

Les variables x_3 et x_5 ont permuté.
On exprime les variables dans la base en fonction des variables hors-base x_5 et x_6.

$$\begin{cases} x_3 = 100 - x_5 + x_6 \\ x_1 = 300 - (100 - x_5 + x_6) = 200 + x_5 - x_6 \\ x_2 = 100 + 2(100 - x_5 + x_6) = 300 - 2x_5 + x_6 \\ x_4 = 300 - 2(100 - x_5 + x_6) = 100 + 2x_5 - x_6 \\ Z = 2600 + 3(100 - x_5 + x_6) = 2900 - 3x_5 - 2x_6 \end{cases} \Leftrightarrow \begin{cases} x_3 + x_5 - x_6 = 100 \\ x_1 - x_5 + x_6 = 200 \\ x_2 + 2x_5 - x_6 = 300 \\ x_4 - 2x_5 + x_6 = 100 \\ Z = 2900 - 3x_5 - 2x_6 \end{cases}$$

On obtient le tableau :

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste	
x_1	1	0	0	0	-1	1	200	$x_1 - x_5 + x_6 = 200$
x_4	0	0	0	1	-2	1	100	$x_4 - 2x_5 + x_6 = 100$
x_3	0	0	1	0	1	-1	100	$x_3 + x_5 - x_6 = 100$
x_2	0	1	0	0	2	-1	300	$x_2 + 2x_5 - x_6 = 300$
Z	0	0	0	0	-3	-2	-2900	$Z = 2900 - 3x_5 - 2x_6$

On a pris la colonne des variables dans la base du troisième tableau et on y a remplacé x_5 par x_3. Pour la fonction économique Z, le coefficient constant 2900 est affecté du signe "-" et on place -2900.

Conclusion :

$$Z = 2900 - 3x_5 - 2x_6,$$

x_5 et x_6 sont hors-base donc nulles, toute augmentation de x_5 ou x_6 entraîne une diminution de Z. Il n'est plus possible d'améliorer la fonction économique, la solution $(x_1 = 200, x_2 = 300)$ est la solution optimale. On interprète les résultats de la manière suivante :

— $x_1 = 200$ bureaux de modèle luxe,
— $x_2 = 300$ bureaux de modèle standard,
— $x_3 = 100$, il reste une possibilité de fabriquer 100 bureaux de modèle luxe,
— $x_4 = 100$, il reste une possibilité de fabriquer 100 bureaux de modèle standard,
— $x_5 = 0$, tout le bois disponible est utilisé,
— $x_6 = 0$, tout le temps disponible est utilisé.
Z est maximum pour $x_1 = 200, x_2 = 300$ et vaut 2900.

Disposition pratique des tableaux

Afin de systématiser et de simplifier les calculs, ceux-ci peuvent être présentés sous forme de tableaux. Un tableau correspond à une solution de base et une itération représente une modification du tableau.
– *Tableau initial*

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste
x_3	1	0	1	0	0	0	300
x_4	0	1	0	1	0	0	400
x_5	1	1	0	0	1	0	500
x_6	2	1	0	0	0	1	700
Z	7	5	0	0	0	0	0

$x_1 = x_2 = 0$ représente le sommet origine et $Z = 0$. On a de plus, $x_3 = 300, x_4 = 400, x_5 = 500$ et $x_6 = 700$.
On sélectionne dans la fonction économique la variable affectée du coefficient strictement positif le plus grand. La variable x_1 entre en base. Quelle est la variable sortante ?
On considère la colonne C obtenue en divisant les coefficients constants par la colonne des coefficients de la variable x_1 qui entre en base.

	x_1	constante	C
x_3	1	300	$\frac{300}{1} = 300$
x_4	0	400	$\frac{400}{0} = +\infty$
x_5	1	500	$\frac{500}{1} = 500$
x_6	2	700	$\frac{700}{2} = 350$

On sélectionne dans cette colonne le plus petit nombre strictement positif 300 . La variable x_3 sort de la base. Les deux variables x_1 et x_3 ont permuté. Le pivot est situé à l'intersection de la colonne

variable entrante et de la ligne variable sortante et est égal à 1 .
Deuxième tableau :
Impérativement dans la colonne des variables dans la base du tableau initial, on remplace la variable x_3 qui sort de la base par la variable x_1 qui entre en base, on recopie les autres variables d'où la disposition du second tableau

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste
x_1							
x_4							
x_5							
x_6							
Z							

Comment remplit-on le tableau ?
On recopie la ligne L_p du pivot (avec un pivot $a = 1$) dans la ligne L_{x_1} :

L_p	1	0	1	0	0	0	300

On doit exprimer le programme en fonction des nouvelles variables hors-base x_2 et x_3.
Pour la ligne L_{x_4},

$$x_2 + x_4 = 400.$$

x_4 s'exprime bien en fonction de x_2 et x_3. On recopie cette ligne.
Pour la ligne L_{x_5},

$$x_1 + x_2 + x_5 = 500.$$

Par une combinaison linéaire de la ligne L_{x_5} et de la ligne pivot L_p, on élimine la variable x_1 qui est entrée en base :

L_{x_5}	1	1	0	0	1	0	500
L_p	1	0	1	0	0	0	300
$L_{x_5} - L_p$	0	1	-1	0	1	0	200

On recopie ensuite cette nouvelle ligne L_{x_5} :

$$x_2 - x_3 + x_5 = 200$$

. Pour la ligne L_{x_6}

L_{x_6}	2	1	0	0	0	1	700
L_p	1	0	1	0	0	0	300
$L_{x_6} - 2L_p$	0	1	-2	0	0	1	100

On recopie cette nouvelle ligne L_{x_6} :

$$x_2 - 2x_3 + x_6 = 100$$

. Pour la ligne de la fonction économique L_Z

L_Z	7	5	0	0	0	0	0
L_p	1	0	1	0	0	0	300
$L_Z - 7L_p$	0	5	-7	0	0	0	-2100

d'où la fonction économique exprimée en fonction des variables hors-base :

$$Z = 5x_2 - 7x_3 + 2100$$

On obtient donc le second tableau :

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste
x_1	1	0	1	0	0	0	300
x_4	0	1	0	1	0	0	400
x_5	0	1	-1	0	1	0	200
x_6	0	1	-2	0	0	1	100
Z	0	5	-7	0	0	0	-2100

On a $x_2 = x_3 = 0$. Comme $x_1 = 300$, on atteint le sommet $A(300, 0)$ et $Z = 2900$. On a de plus, $x_4 = 400, x_5 = 200, x_6 = 100$.

— *Troisième tableau :*

$$Z = 5x_2 - 7x_3 + 2100,$$

la variable entrante est x_2 (une augmentation de x_3 entraîne une dimimution de Z). Déterminons la variable sortante : la colonne C est donnée par :

	x_2	constante	C
x_1	0	300	$\frac{300}{0} = +\infty$
x_4	1	400	$\frac{400}{1} = 400$
x_5	1	200	$\frac{200}{1} = 200$
x_6	1	100	$\frac{100}{1} = 100$

On sélectionne dans cette colonne C le coefficient strictement positif le plus petit c'est-à-dire 100, la variable x_6 sort de la base. Les variables x_2 et x_6 ont permuté. Le pivot est situé à l'intersection de la colonne variable entrante et de la ligne variable sortante. Ce pivot vaut 1.

On remplit le troisième tableau : dans la colonne des variables dans la base du deuxième tableau, on remplace la variable x_6 qui sort de la base par la variable x_2 qui entre en base. On recopie la ligne pivot avec le pivot de 1 :

$$L_p : 1.x_2 - 2.x_3 + 1.x_6 = 100$$

VHB / VDB	x_1	x_2	x_3	x_4 •	x_5 •	x_6	cste
x_1							
x_4							
x_5							
x_2	0	1	-2	0	0	1	100
Z							

Par des combinaisons avec la ligne pivot, on exprime le système en fonction des variables hors-base x_3 et x_6 c'est-à-dire qu'on élimine la variable x_2 qui est entrée en base :

— pour la ligne L_{x_4} :

L_{x_4}	0	1	0	1	0	0	400
L_p	0	1	-2	0	0	1	100
$L_{x_4} - L_p$	0	0	2	1	0	-1	300

— pour la ligne L_{x_5} :

L_{x_5}	0	1	-1	0	1	0	200
L_p	0	1	-2	0	0	1	100
$L_{x_5} - L_p$	0	0	1	0	1	-1	100

— pour la ligne L_Z :

L_Z	0	5	-7	0	0	0	-2100
L_p	0	1	-2	0	0	1	100
$L_Z - 5L_p$	0	0	3	0	0	-5	-2600

Une fois le tableau rempli, on obtient :

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste
x_1	1	0	1	0	0	0	300
x_4	0	0	2	1	0	-1	300
x_5	0	0	1	0	1	-1	100
x_2	0	1	-2	0	0	1	100
Z	0	0	3	0	0	-5	-2600

On a $x_3 = x_6 = 0$. Comme $x_1 = 300$ et $x_2 = 100$, on est passé du sommet $A(300, 0)$ au sommet $B(300, 100)$. On a de plus $x_4 = 300, x_5 = 100$ et Z vaut 2600 .

— *Quatrième tableau :*

$$Z = 3x_3 - 5x_6 + 2600$$

La variable entrante est x_3 (toute augmentation de x_6 entraîne une diminution de Z). Déterminons la variable sortante : la colonne C est donnée par

	x_3	constante	C
x_1	1	300	$\frac{300}{1} = 300$
x_4	2	300	$\frac{300}{2} = 150$
x_5	1	100	$\frac{100}{1} = 100$
x_2	-2	100	$\frac{100}{-2} = -50$

On sélectionne dans cette colonne C le coefficient strictement positif le plus petit c'est-à-dire 100, la variable x_5 sort de la base. Les variables x_3 et x_5 ont permuté. Le pivot est 1, il est situé à l'intersection de la colonne x_3 et de la ligne x_5. On remplit le quatrième tableau : dans la colonne des variables dans la base du troisième tableau, on remplace la variable x_5 qui sort de base par la variable x_3 qui entre en base. On recopie la ligne pivot avec un pivot de 1 :
$L_p : x_3 + x_5 - x_6 = 100$

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste
x_1							
x_4							
x_3	0	0	1	0	1	-1	100
x_2							
Z							

Par des combinaisons avec la ligne pivot, on exprime le système en fonction des variables hors-base x_5 et x_6 c'est-à-dire qu'on élimine la variable x_3 qui est entrée en base :

— pour la ligne L_{x_1} :

L_{x_1}	1	0	1	0	0	0	300
L_p	0	0	1	0	1	-1	100
$L_{x_1} - L_p$	1	0	0	0	-1	1	200

. pour la ligne L_{x_4} :

L_{x_4}	0	0	2	1	0	-1	300
L_p	0	0	1	0	1	-1	100
$L_{x_4} - 2L_p$	0	0	0	1	-2	1	100

— pour la ligne L_{x_2} :

L_{x_2}	0	1	-2	0	0	1	100
L_p	0	0	1	0	1	-1	100
$L_{x_2} + 2L_p$	0	1	0	0	2	-1	300

— pour la ligne L_Z :

L_Z	0	0	3	0	0	-5	-2600
L_p	0	0	1	0	1	-1	100
$L_Z - 3L_p$	0	0	0	0	-3	-2	-2900

On peut ensuite remplir le quatrième tableau :

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste
x_1	1	0	0	0	-1	1	200
x_4	0	0	0	1	-2	1	100
x_3	0	0	1	0	1	-1	100
x_2	0	1	0	0	2	-1	300
Z	0	0	0	0	-3	-2	-2900

Conclusion, la fonction économique s'écrit $Z = 2900 - 3x_5 - 2x_6$ où x_5 et x_6 sont les variables hors-base donc nulles. Toute augmentation de x_5 et x_6 conduit à une diminution de Z. Donc $x_1 = 200, x_2 = 300, x_3 = 100$, $x_4 = 100$ et $Z = 2900$. La fonction économique atteint son maximum au point $C(200, 300)$ et vaut 2900.
Le pseudocode suivant décrit cet algorithme :

Algorithm 1 Algorithme du simplexe

Entrée : un programme linéaire en forme standard :

$$\max\ c^T x \quad \text{s.c. } Ax = b,\ x \geq 0$$

avec $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$
Sortie : une solution optimale x^* ou détection de non-bornitude
Initialiser une solution de base réalisable x^0 associée à une base B
Tant que la solution actuelle nest pas optimale **faire**
    Calculer les coûts réduits $c_j - c_B^T A_B^{-1} A_j$ pour toutes les variables hors base j
    Si tous les coûts réduits ≤ 0 **alors**
        Retourner la solution courante x comme optimale
    Fin Si
    Choisir une variable entrante x_k avec coût réduit > 0
    Calculer la direction $d = A_B^{-1} A_k$
    Si $d \leq 0$ **alors**
        Le problème est non borné (pas de solution optimale finie)
    Fin Si
    Calculer le pas $\theta = \min\{x_B i/d_i \mid d_i > 0\}$
    Identifier la variable sortante x_r correspondante
    Mettre à jour la base $B \leftarrow (B \setminus \{r\}) \cup \{k\}$
    Mettre à jour la solution de base x
Fin Tant que

Exemple 1.2.5 *On considère le programme linéaire suivant*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \\ x_1 + 3x_2 + 2x_3 \leq 40 \\ 3x_1 + 2x_2 + x_3 \leq 45 \\ x_1 + x_2 + 4x_3 \leq 38 \\ Z(x_1, x_2, x_3) = 10x_1 + 14x_2 + 12x_3 \ \textit{à maximiser} \end{cases}$$

1. *Programme standard :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0 \\ x_1 + 3x_2 + 2x_3 + x_4 = 40 \\ 3x_1 + 2x_2 + x_3 + x_5 = 45 \\ x_1 + x_2 + 4x_3 + x_6 = 38 \\ Z(x_1, x_2, x_3) = 10x_1 + 14x_2 + 12x_3 \text{ à maximiser} \end{cases}$$

   La solution de base de départ du programme correspond au sommet 0, c'est la solution nulle qui consiste à ne rien produire : $x_1 = x_2 = x_3 = 0$ et $Z(0, 0, 0) = 0$. Les variables x_1, x_2, x_3 sont hors-base donc nulles, les autres variables x_4, x_5, x_6 sont dans la base.
   W

2. *Tableau initial :*

VHB / VDB	x_1	x_2	x_3	x_4 •	x_5 •	x_6 •	cste
x_4	1	3	2	1	0	0	40
x_5	3	2	1	0	1	0	45
x_6	1	1	4	0	0	1	38
Z	10	14	12	0	0	0	0

$$x_1 + 3x_2 + 2x_3 + x_4 = 40$$
$$3x_1 + 2x_2 + x_3 + x_5 = 45$$
$$x_1 + x_2 + 4x_3 + x_6 = 38$$
$$Z = 10x_1 + 14x_2 + 12x_3$$

- Choix de la variable entrante : on sélectionne la variable affectée du coefficient strictement positif le plus grand dans la fonction économique, la variable x_2 entre en base.
- Choix de la variable sortante : on détermine la colonne C :

	x_2	constante	C
x_4	3	40	$\frac{40}{3} \simeq 13,33$
x_5	2	45	$\frac{45}{2} = 22,5$
x_6	1	38	$\frac{38}{1} = 38$

On sélectionne le coefficient strictement positif le plus petit dans la colonne C, la variable x_4 sort de la base.

VDB	VHB
x_2	x_4
x_5	x_1
x_6	x_3

- Le pivot : il est situé à l'intersection de la colonne variable qui entre en base et de la ligne variable qui sort de la base, ce pivot est 3 . Afin d'obtenir un pivot de 1, on divise tous les coefficients de la ligne pivot par ce pivot 3 . On obtient la nouvelle ligne pivot :

$$L_p : \frac{1}{3}x_1 + x_2 + \frac{2}{3}x_3 + \frac{1}{3}x_4 = \frac{40}{3}$$

soit

L_p	$\frac{1}{3}$	1	$\frac{2}{3}$	1	0	0	$\frac{40}{3}$

3. *Deuxième tableau :*

VHB / VDB	x_1	x_2 •	x_3	x_4	x_5 •	x_6 •	cste
x_2	$\frac{1}{3}$	1	$\frac{2}{3}$	$\frac{1}{3}$	0	0	$\frac{40}{3}$
x_5							
x_6							
Z							

- On recopie la ligne du pivot avec le pivot de 1 .

- Pour remplir ce second tableau, par des combinaisons avec la ligne pivot, on élimine la variable x_2 qui est entrée en base :
  - ∗ pour la ligne L_{x_5} :

L_{x_5}	3	2	1	0	1	0	45
L_p	$\frac{1}{3}$	1	$\frac{2}{3}$	$\frac{1}{3}$	0	0	$\frac{40}{3}$
$L_{x_5} - 2L_p$	$\frac{7}{3}$	0	$-\frac{1}{3}$	$-\frac{2}{3}$	1	0	$\frac{55}{3}$

  - ∗ pour la ligne L_{x_6} :

L_{x_6}	1	1	4	0	0	1	38
L_p	$\frac{1}{3}$	1	$\frac{2}{3}$	$\frac{1}{3}$	0	0	$\frac{40}{3}$
$L_{x_6} - L_p$	$\frac{2}{3}$	0	$\frac{10}{3}$	$-\frac{1}{3}$	0	1	$\frac{74}{3}$

  - ∗ pour la ligne L_Z :

L_Z	10	14	12	0	0	0	0
L_p	$\frac{1}{3}$	1	$\frac{2}{3}$	$\frac{1}{3}$	0	0	$\frac{40}{3}$
$L_Z - 14L_p$	$\frac{16}{3}$	0	$\frac{8}{3}$	$-\frac{14}{3}$	0	0	$-\frac{560}{3}$

  Le deuxième tableau s'écrit alors :

VHB / VDB	x_1	x_2 •	x_3	x_4	x_5 •	x_6 •	cste
x_2	$\frac{1}{3}$	1	$\frac{2}{3}$	$\frac{1}{3}$	0	0	$\frac{40}{3}$
x_5	$\frac{7}{3}$	0	$-\frac{1}{3}$	$-\frac{2}{3}$	1	0	$\frac{55}{3}$
x_6	$\frac{2}{3}$	0	$\frac{10}{3}$	$-\frac{1}{3}$	0	1	$\frac{74}{3}$
Z	$\frac{16}{3}$	0	$\frac{8}{3}$	$-\frac{14}{3}$	0	0	$-\frac{560}{3}$

  On a par conséquent

$$Z = \frac{16}{3}x_1 + \frac{8}{3}x_3 - \frac{14}{3}x_4 + \frac{560}{3}$$

4. *Troisième tableau :*
   - La variable entrante est x_1 ; en effet, $\frac{16}{3}$ est le coefficient strictement positif le plus grand dans la fonction économique.
   - La variable sortante est déterminée à l'aide de la colonne C :

	x_1	constante	C
x_2	$\frac{1}{3}$	$\frac{40}{3}$	$\frac{40}{3}/\frac{1}{3} = 40$
x_5	$\frac{7}{3}$	$\frac{55}{3}$	$\frac{55}{3}/\frac{7}{3} = \frac{55}{7}$
x_6	$\frac{2}{3}$	$\frac{74}{3}$	$\frac{74}{3}/\frac{2}{3} = 37$

     On choisit le coefficient strictement positif le plus petit dans la colonne C soit $\frac{55}{7}$, la variable x_5 sort de la base.
   - Le pivot est $\frac{7}{3}$, situé à l'intersection de la colonne variable qui entre en base et de la ligne variable qui sort de la base. Pour obtenir un pivot de 1, on divise la ligne pivot par ce pivot $\frac{7}{3}$, on obtient
     soit

$$L_p = x_1 - \frac{1}{7}x_3 - \frac{2}{7}x_4 + \frac{3}{7}x_5 = \frac{55}{7}$$

L_p	1	0	$-\frac{1}{7}$	$-\frac{2}{7}$	$\frac{3}{7}$	0	$\frac{55}{7}$

- On exprime le système en fonction des nouvelles variables hors-base x_3 et x_4 et on élimine x_1 qui est entrée en base.
  - ∗ Pour la ligne L_{x_2} :

L_{x_2}	$\frac{1}{3}$	1	$\frac{2}{3}$	$\frac{1}{3}$	0	0	$\frac{40}{3}$
L_p	1	0	$-\frac{1}{7}$	$-\frac{2}{7}$	$\frac{3}{7}$	0	$\frac{55}{7}$
$L_{x_2} - \frac{1}{3}L_p$	0	1	$\frac{5}{7}$	$\frac{3}{7}$	$-\frac{1}{7}$	0	$\frac{75}{7}$

  - ∗ Pour la ligne L_{x_6} :

L_{x_6}	$\frac{2}{3}$	0	$\frac{10}{3}$	$-\frac{1}{3}$	0	1	$\frac{74}{3}$
L_p	1	0	$-\frac{1}{7}$	$-\frac{2}{7}$	$\frac{3}{7}$	0	$\frac{55}{7}$
$L_{x_6} - \frac{2}{3}L_p$	0	0	$\frac{24}{7}$	$-\frac{1}{7}$	$-\frac{2}{7}$	1	$\frac{136}{7}$

  - ∗ Pour la ligne L_Z :

L_Z	$\frac{16}{3}$	0	$\frac{8}{3}$	$-\frac{14}{3}$	0	0	$-\frac{560}{3}$
L_p	1	0	$-\frac{1}{7}$	$-\frac{2}{7}$	$\frac{3}{7}$	0	$\frac{55}{7}$
$L_Z - \frac{16}{3}L_p$	0	0	$\frac{24}{7}$	$-\frac{22}{7}$	$-\frac{16}{7}$	0	$-\frac{1600}{7}$

  On peut maintenant remplir le troisième tableau :

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste
x_2	0	1	$\frac{5}{7}$	$\frac{3}{7}$	$-\frac{1}{7}$	0	$\frac{75}{7}$
x_1	1	0	$-\frac{1}{7}$	$-\frac{2}{7}$	$\frac{3}{7}$	0	$\frac{55}{7}$
x_6	0	0	$\frac{24}{7}$	$-\frac{1}{7}$	$-\frac{2}{7}$	1	$\frac{136}{7}$
Z	0	0	$\frac{24}{7}$	$-\frac{22}{7}$	$-\frac{16}{7}$	0	$-\frac{1600}{7}$

5. *Quatrième tableau :*

$$Z = \frac{24}{7}x_3 - \frac{22}{7}x_4 - \frac{16}{7}x_5 + \frac{1600}{7}$$

- Variable entrante : on sélectionne le coefficient $\frac{24}{7}$, la variable x_3 entre en base.
- Variable sortante :

	x_3	Constante	C
x_2	$\frac{5}{7}$	$\frac{75}{7}$	$\frac{75}{7}/\frac{5}{7} = 15$
x_1	$-\frac{1}{7}$	$\frac{55}{7}$	$\frac{55}{7}/-\frac{1}{7} = -55$
x_6	$\frac{24}{7}$	$\frac{136}{7}$	$\frac{136}{7}/\frac{24}{7} = \frac{17}{3}$

  La variable x_6 sort de base.
- Le pivot est $\frac{24}{7}$, on divise la ligne pivot par ce pivot et on obtient la nouvelle ligne pivot :

$$L_p : x_3 - \frac{1}{24}x_4 - \frac{1}{12}x_5 + \frac{7}{24}x_6 = \frac{17}{3}$$

L_p	0	0	1	$-\frac{1}{24}$	$-\frac{1}{12}$	$\frac{7}{24}$	$\frac{17}{3}$

  Dans la colonne variables dans la base du troisième tableau, on remplace la variable x_6 par la variable x_3 et on y recopie la nouvelle ligne pivot

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste
x_2							
x_1							
x_3	0	0	1	$-\frac{1}{24}$	$-\frac{1}{12}$	$\frac{7}{24}$	$\frac{17}{3}$
Z							

On exprime le système en fonction des variables hors-base x_4, x_5 et x_6. On élimine la variable x_3 qui est entrée en base :

* pour la ligne L_{x_2} :

L_{x_2}	0	1	$\frac{5}{7}$	$\frac{3}{7}$	$-\frac{1}{7}$	0	$\frac{75}{7}$
L_p	0	0	1	$-\frac{1}{24}$	$-\frac{1}{12}$	$\frac{7}{24}$	$\frac{17}{3}$
$L_{x_2} - \frac{5}{7}L_p$	0	1	0	$\frac{11}{24}$	$-\frac{1}{12}$	$-\frac{5}{24}$	$\frac{20}{3}$

* pour la ligne L_{x_1} :

L_{x_1}	1	0	$-\frac{1}{7}$	$-\frac{2}{7}$	$\frac{3}{7}$	0	$\frac{55}{7}$
L_p	0	0	1	$-\frac{1}{24}$	$-\frac{1}{12}$	$\frac{7}{24}$	$\frac{17}{3}$
$L_{x_1} + \frac{1}{7}L_p$	1	0	0	$-\frac{7}{24}$	$\frac{5}{12}$	$\frac{1}{24}$	$\frac{26}{3}$

* pour la ligne L_Z :

L_Z	0	0	$\frac{24}{7}$	$-\frac{22}{7}$	$-\frac{16}{7}$	0	$-\frac{1600}{7}$
L_p	0	0	1	$-\frac{1}{24}$	$-\frac{1}{12}$	$\frac{7}{24}$	$\frac{17}{3}$
$L_Z - \frac{24}{7}L_p$	0	0	0	-3	-2	-1	-248

Le quatrième tableau est finalement donné par :

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste
x_2	0	1	0	$\frac{11}{24}$	$-\frac{1}{12}$	$-\frac{5}{24}$	$\frac{20}{3}$
x_1	1	0	0	$-\frac{7}{24}$	$\frac{5}{12}$	$\frac{1}{24}$	$\frac{26}{3}$
x_3	0	0	1	$-\frac{1}{24}$	$-\frac{1}{12}$	$\frac{7}{24}$	$\frac{17}{3}$
Z	0	0	0	-3	-2	-1	-248

6. *Conclusion :*

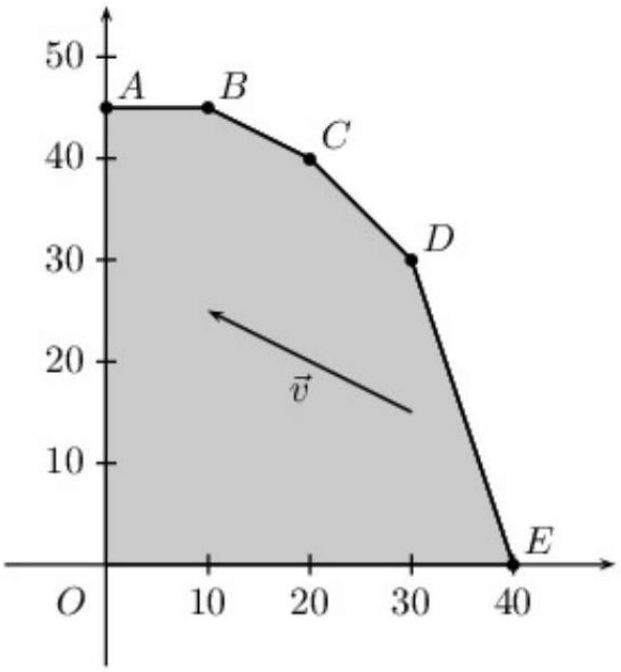
$$Z = -3x_4 - 2x_5 - x_6 + 248$$

Les trois variables x_4, x_5 et x_6 sont affectées de coefficients négatifs, toute augmentation de x_4, x_5 ou x_6 diminuerait la valeur de Z. Il n'est plus possible d'améliorer la fonction économique.
Z est maximum pour $x_4 = 0, x_5 = 0, x_6 = 0, x_1 = \frac{20}{3}, x_2 = \frac{26}{3}, x_3 = \frac{17}{3}$, atteint son maximum au point $\left(\frac{20}{3}, \frac{26}{3}, \frac{17}{3}\right)$ et vaut $Z\left(\frac{20}{3}, \frac{26}{3}, \frac{17}{3}\right) = 248$. De plus, comme $x_4 = 0, x_5 = 0$ et $x_6 = 0$, les trois matières premières sont utilisées en totalité.

Exemple 1.2.6 *(infinité de solutions)*
On se donne le programme linéaire suivant :

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_2 \leq 45 \\ 3x_1 + x_2 \leq 120 \\ x_1 + 2x_2 \leq 100 \\ x_1 + x_2 \leq 60 \\ \quad \textit{Maximiser } Z(x_1, x_2) = x_1 + 2x_2 \end{cases}$$

1. *Résolution graphique*
Les côtés du polygone sont définis de la manière suivante :
— le segment $[AB] : x_2 = 45, 0 \leq x_1 \leq 10$



— le segment $[BC] : x_1 + 2x_2 = 100, 10 \leq x_1 \leq 20$

— le segment $[CD] : x_1 + x_2 = 60, 20 \leq x_1 \leq 30$

— le segment $[DE] : 3x_1 + x_2 = 60, 30 \leq x_1 \leq 40$

On trace les droites D_Z d'équations $Z = x_1 + 2x_2$, ces droites ont pour vecteur directeur $\vec{v}\binom{-2}{1}$. Elles sont parallèles entre-elles et de plus, elles sont parallèles au côté (BC) d'équation $x_1 + 2x_2 = 100$, $10 \leq x_1 \leq 20$. La droite D_Z qui rencontre le domaine des solutions réalisables et qui a une ordonnée à l'origine maximale est la droite qui contient B et C, d'équation $x_1 + 2x_2 = 100$. La fonction économique atteint son maximum 100 en tous les points du segment $[BC] : \begin{cases} x_1 + 2x_2 = 100 \\ 10 \leq x_1 \leq 20. \end{cases}$

2. *Le simplexe*
Comment fait-on apparaître cette infinité de couples solutions dans les tableaux du simplexe ?
(a) Tableau initial : les variables hors-base sont x_1 et x_2, le programme standard est donné par :

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0 \\ x_2 + x_3 = 45 \\ 3x_1 + x_2 + x_4 = 120 \\ x_1 + 2x_2 + x_5 = 100 \\ x_1 + x_2 + x_6 = 60 \\ Z(x_1, x_2) = x_1 + 2x_2 \text{ à maximiser} \end{cases}$$

Le tableau initial peut s'écrire sous la forme :

VHB / VDB	x_1	x_2	x_3 •	x_4 •	x_5 •	x_6 •	cste	C
x_3	0	1	1	0	0	0	45	$\frac{45}{1} = 45$
x_4	3	1	0	1	0	0	120	$\frac{120}{1} = 120$
x_5	1	2	0	0	1	0	100	$\frac{100}{2} = 50$
x_6	1	1	0	0	0	1	60	$\frac{60}{1} = 60$
Z	1	2						

(b) *Première itération*
On sélectionne la variable x_2 qui entre en base, la variable qui sort de base est x_3. Le pivot est 1, la ligne pivot est

L_p	0	1	1	0	0	0	45

les variables x_2 et x_3 permutent
. Pour la ligne L_{x_4} :

L_{x_4}	3	1	0	1	0	0	120
L_p	0	1	1	0	0	0	45
$L_{x_4} - L_p$	3	0	-1	1	0	0	75

. Pour la ligne L_{x_5} :

L_{x_5}	1	2	0	0	1	0	100
L_p	0	1	1	0	0	0	45
$L_{x_5} - 2L_p$	1	0	-2	0	1	0	10

. Pour la ligne L_{x_6} :

L_{x_6}	1	1	0	0	0	1	60
L_p	0	1	1	0	0	0	45
$L_{x_6} - L_p$	1	0	-1	0	0	1	15

. Pour la ligne L_Z :

L_Z	1	2					
L_p	0	1	1	0	0	0	45
$L_Z - 2L_p$	1	0	-2	0	0	0	-90

VHB / VDB	x_1	x_2 •	x_3	x_4 •	x_5 •	x_6 •	cste	C
x_2	0	1	1	0	0	0	45	$\frac{45}{0} = +\infty$
x_4	3	0	-1	1	0	0	75	$\frac{75}{3} = 25$
x_5	1	0	-2	0	1	0	10	$\frac{10}{1} = 10$
x_6	1	0	-1	0	0	1	15	$\frac{15}{1} = 15$
Z	1	0	-2	0	0	0	-90	

Cette première itération conduit du sommet $O(0,0)$ au sommet $A(0,45)$ et $Z(0,45) = 90$.
(c) Deuxième itération

$$Z = x_1 - 2x_3 + 90,$$

la variable x_1 entre en base, la variable x_5 sort de base, le pivot est 1, la ligne pivot est

$$L_p : x_1 - 2x_3 + x_5 = 10$$

soit

L_p	1	0	-2	0	1	0	10

les variables x_1 et x_5 permutent

L_{x_2}	0	1	1	0	0	0	45
L_p	1	0	-2	0	1	0	10
L_{x_2}	0	1	1	0	0	0	45

L_{x_4}	3	0	-1	1	0	0	75
L_p	1	0	-2	0	1	0	10
$L_{x_4} - 3L_p$	0	0	5	1	-3	0	45

L_{x_6}	1	0	-1	0	0	1	15
L_p	1	0	-2	0	1	0	10
$L_{x_6} - L_p$	0	0	1	0	-1	1	5

L_Z	1	0	-2	0	0	0	-90
L_p	1	0	-2	0	1	0	10
$L_Z - L_p$	0	0	0	0	-1	0	-100

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste	C
x_2	0	1	1	0	0	0	45	$\frac{45}{1} = 45$
x_4	0	0	5	1	-3	0	45	$\frac{45}{5} = 9$
x_1	1	0	-2	0	1	0	10	$\frac{10}{-2} = -5$
x_6	0	0	1	0	-1	1	5	$\frac{5}{1} = 5$
Z	0	0	0	0	-1	0	-100	

Les variables hors-base sont x_3, x_5 et

$$Z = 0.x_3 + (-1).x_5 + 100 = -x_5 + 100.$$

Pour $x_3 = x_5 = 0$, on obtient $x_1 = 10, x_2 = 45, x_4 = 45$ et $x_6 = 5$. On atteint le sommet $B(10,45)$. Dans la fonction économique, la variable hors-base x_3 est affectée du coefficient 0 . Si on augmente x_3, Z sera invariant et égal à 100 .
(d) *Troisième itération*
On fait entrer en base x_3, x_6 sort de base, le pivot est 1, la ligne pivot est

$$L_p : x_3 - x_5 + x_6 = 5$$

soit

L_p	0	0	1	0	-1	1	5

x_3 et x_6 ont permuté

L_{x_2}	0	1	1	0	0	0	45
L_p	0	0	1	0	-1	1	5
$L_{x_2} - L_p$	0	1	0	0	1	-1	40

L_{x_4}	0	0	5	1	-3	0	45
L_p	0	0	1	0	-1	1	5
$L_{x_4} - 5L_p$	0	0	0	1	2	-5	20

L_{x_1}	1	0	-2	0	1	0	10
L_p	0	0	1	0	-1	1	5
$L_{x_1} + 2L_p$	1	0	0	0	-1	2	20

L_Z	0	0	0	0			
L_p	0	0	1	0	-1	1	5
$L_Z + 0.L_p$	0	0	0	0	-1	0	-100

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste
x_2	0	1	0	0	1	-1	40
x_4	0	0	0	1	2	-5	20
x_1	1	0	0	0	-1	2	20
x_3	0	0	1	0	-1	1	5
Z	0	0	0	0	-1	0	-100

La fonction économique s'écrit :

$$Z = -1.x_5 + 0.x_6 + 100 = -x_5 + 100.$$

Les variables hors-base sont x_5, x_6. Pour $x_5 = x_6 = 0$, on obtient $x_1 = 20, x_2 = 40, x_3 = 5$, $x_4 = 20$ et $Z = 100$.Z est maximum pour le deuxième sommet $C(20, 40)$. On a obtenu Z maximum pour deux sommets adjacents $B(10, 45)$ et $C(20, 40)$. On admettra que la fonction économique atteint son maximum en tous les points du segment $[BC]$.

Remarque 1.2.7

1. La présence d'un zéro dans la ligne pivot entraîne l'invariance de la colonne correspondante.

Reprenons le tableau initial de l'exemple (1.2.4)

VHB / VDB	x_1	x_2	x_3 •	x_4 •	x_5 •	x_6 •	cste	C
x_3	1	0	1	0	0	0	300	$\frac{300}{1} = 300$
x_4	0	1	0	1	0	0	400	$\frac{400}{0} = +\infty$
x_5	1	1	0	0	1	0	500	$\frac{500}{1} = 500$
x_6	2	1	0	0	0	1	700	$\frac{700}{2} = 350$
Z	7	5	0	0	0	0	0	

Le pivot est 1, dans la ligne pivot, les variables x_2, x_4, x_5, x_6 sont affectées du coefficient 0 . Ces quatre colonnes seront invariantes dans le tableau suivant, on peut donc recopier ces quatre colonnes sans effectuer de calculs.

VHB / VDB	x_1	x_2	x_3	x_4 •	x_5 •	x_6 •	cste
x_1	1	0	1	0	0	0	300
x_4	▭	1	▭	1	0	0	
x_5	▭	1		0	1	0	
x_6	▭	1		0	0	1	
Z	▭	5		0	0	0	

On peut aussi recopier la ligne pivot. De plus, le système doit s'exprimer en fonction des variables hors-base x_2, x_3 donc x_4 s'exprime en fonction de x_2, x_3 ainsi que x_5, x_6 et Z d'où les compléments dans le tableau "encadrés". On peut donc, sans effectuer de calculs, remplir certaines cases du tableau.
2. La présence d'un zéro dans la colonne du pivot entraîne l'invariance de la ligne correspondante. On reprend le second tableau de l'exemple (1.2.4). En utilisant les deux remarques 1. et 2., on obtient

VHB / VDB	x_1	x_2	x_3	x_4 •	x_5 •	x_6 •	cste	C
x_1	1	0	1	0	0	0	300	$\frac{300}{1} = 300$
x_4	0	1	0	1	0	0	400	$\frac{400}{1} = 400$
x_5	0	1	-1	0	1	0	200	$\frac{200}{1} = 200$
x_6	0	1	-2	0	0	1	100	$\frac{100}{1} = 100$
Z	0	5	-7	0	0	0	-200	

x_2 entre en base, x_6 sort de base, le pivot est 1, la ligne pivot est

L_p	0	1	-2	0	0	1	100

. Dans la ligne du pivot, les variables x_1, x_4 ou x_5 sont affectées du coefficient 0, on recopiera ces trois colonnes.
. On recopie la ligne du pivot.
. Dans la colonne du pivot apparaît un zéro, on recopie la ligne L_{x_1}.
Ces remarques permettent donc d'obtenir deux lignes et trois colonnes du tableau suivant

VHB / VDB	x_1	x_2	x_3	x_4 •	x_5 •	x_6 •	cste
x_1	1	0	1	0	0	0	300
x_4	0			1	0	▭	
x_5	0			0	1	▭	
x_2	0	1	-2	0	0	1	100
Z	0			0	0	▭	

Les variables hors-base étant x_2, x_3, le système s'écrit en fonction des variables hors-base seulement d'où les compléments dans le tableau "encadrés".
Il reste neuf cases à remplir dans le tableau.

3. Si deux coefficients positifs dans la fonction économique sont égaux, on pourra déterminer dans chaque colonne correspondante le pivot éventuel et le rapport associé. On choisira comme pivot celui qui correspond au plus grand rapport.

Exemple 1.2.8 *On se donne le programme linéaire suivant :*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \\ 3x_1 + 5x_2 + x_3 \leq 150 \\ x_1 + 4x_2 + 2x_3 + x_5 \leq 80 \\ Z(x_1, x_2, x_3) = 2x_1 + 2x_2 + x_3 \text{ à maximiser} \end{cases}$$

Le programme standard s'écrit

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0 \\ 3x_1 + 5x_2 + x_3 + x_4 = 150 \\ x_1 + 4x_2 + 2x_3 + x_5 = 80 \\ Z(x_1, x_2, x_3) = 2x_1 + 2x_2 + x_3 \text{ à maximiser} \end{cases}$$

VHB / VDB	x_1	x_2	x_3	x_4 •	x_5 •	cste
x_4	3	5	1	1	0	150
x_5	1	4	2	0	1	80
Z	2	2	1			

1. Si l'on choisit comme variable sortante x_1, la colonne C est alors

	x_1		C
x_4	3	150	$\frac{150}{3} = 50$
x_5	1	80	$\frac{80}{1} = 80$

La variable sortante est x_4, le pivot est égal à 3, le rapport vaut 50 .
2. Si l'on choisit comme variable sortante x_2, la colonne C est donnée par :

	x_2		C
x_4	5	150	$\frac{150}{5} = 30$
x_5	4	80	$\frac{80}{4} = 20$

La variable sortante est alors x_5, le pivot est égal à 4 et le rapport vaut 20 .
On choisit comme variable sortante celle qui correspond au plus grand rapport. Dans l'exemple, $50 > 20$, la variable sortante est x_4, la variable entrante x_1, le pivot est 3.
La règle d'entrée du plus grand gain marginal nous propose une méthode qui permet d'obtenir la valeur optimale de Z, mais rien n'indique que cette méthode propose le plus court chemin.

Exemple 1.2.9 *Soit le programme linéaire*

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \\ x_1 \leq 5 \\ 4x_1 + x_2 \leq 25 \\ 8x_1 + 4x_2 + x_3 \leq 125 \\ Z(x_1, x_2, x_3) = 4x_1 + 2x_2 + x_3 \text{ à maximiser} \end{cases}$$

Le programme standard s'écrit

$$\begin{cases} x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0 \\ x_1 + x_4 = 5 \\ 4x_1 + x_2 + x_5 = 25 \\ 8x_1 + 4x_2 + x_3 + x_6 = 125 \\ Z(x_1, x_2, x_3) = 4x_1 + 2x_2 + x_3 \text{ à maximiser} \end{cases}$$

On a le tableau :

VHB / VDB	x_1	x_2	x_3	x_4 •	x_5 •	x_6 •	cste	C
x_4	1	0	0	1	0	0	5	5
x_5	4	1	0	0	1	0	25	$\frac{25}{4} = 6,25$
x_6	8	4	1	0	0	1	125	$\frac{125}{8} = 15,625$
Z	4	2	1	0	0	0	0	

On se trouve au sommet origine $O(0,0), x_4 = 5, x_5 = 25, x_6 = 125$ et $Z = 0$.
On applique la règle du plus grand gain marginal, x_1 entre en base, x_4 sort de base, le pivot est 1 .
On obtient le tableau suivant

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste	C
x_4	1	0	0	1	0	0	5	$\frac{5}{0} = +\infty$
x_5	0	1	0	-4	1	0	5	$\frac{5}{1} = 5$
x_6	0	4	1	-8	0	1	85	$\frac{85}{4} = 21,25$
Z	0	2	1	-4	0	0	-20	

On se trouve au sommet A_1 de coordonnées $(5,0,0)$ avec $x_4 = 0, x_5 = 5, x_6 = 85$ et $Z = 20$.
x_2 entre en base, x_5 sort de base, le pivot est 1.

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6 •	cste	C
x_1	1	0	0	1	0	0	5	$\frac{5}{1} = 5$
x_2	0	1	0	-4	1	0	5	$-\frac{5}{4}$
x_6	0	0	1	8	-4	1	65	$\frac{65}{8} = 8,125$
Z	0	0	1	4	-2	0	-30	

On se trouve au sommet A_2 de coordonnées $(5,5,0)$ avec $x_4 = 0, x_5 = 0, x_6 = 65$ et $Z = 30$.
x_4 entre en base, x_1 sort de base, le pivot est 1.

VHB / VDB	x_1	x_2 •	x_3	x_4 •	x_5	x_6 •	cste	C
x_4	1	0	0	1	0	0	5	$+\infty$
x_2	4	1	0	0	1	0	25	$+\infty$
x_6	-8	0	1	0	-4	1	25	25
Z	-4	0	1	0	-2	0	-50	

On se trouve au sommet A_3 de coordonnées $(0,25,0)$ avec $x_4 = 5, x_5 = 0, x_6 = 25$ et $Z = 50$.
x_3 entre en base, x_6 sort de base, le pivot est 1.

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste	C
x_4	1	0	0	1	0	0	5	5
x_2	4	1	0	0	1	0	25	6,25
x_3	-8	0	1	0	-4	1	25	-3,125
Z	4	0	0	0	-2	-1	-75	

On se trouve au sommet A_4 de coordonnées $(0,25,25)$ avec $x_4 = 5, x_5 = 0, x_6 = 0$ et $Z = 75$.
x_1 entre en base, x_4 sort de base, le pivot est 1.

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste	C
x_1	1	0	0	1	0	0	5	$+\infty$
x_2	0	1	0	-4	1	0	5	5
x_3	0	0	1	8	-4	1	65	-16,25
Z	0	0	0	-4	2	-1	-95	

On se trouve au sommet A_5 de coordonnées $(5, 5, 65)$ avec $x_4 = 0, x_5 = 0, x_6 = 0$ et $Z = 95$.
x_5 entre en base, x_2 sort de base, le pivot est 1.

VHB / VDB	x_1	x_2	x_3 •	x_4	x_5 •	x_6	cste	C
x_1	1	0	0	1	0	0	5	5
x_5	0	1	0	-4	1	0	5	-1,25
x_3	0	4	1	-8	0	1	85	-10,625
Z	0	-2	0	4	0	-1	-105	

On se trouve au sommet A_6 de coordonnées $(5, 0, 85)$ avec $x_4 = 0, x_5 = 5, x_6 = 0$ et $Z = 105$. x_4 entre en base, x_1 sort de base, le pivot est 1.

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	C
x_4	1	0	0	1	0	0	5
x_5	4	1	0	0	1	0	25
x_3	8	4	1	0	0	1	125
Z	-4	-2	0	0	0	-1	-125

On se trouve au sommet A_7 de coordonnées $(0, 0, 125)$ avec $x_4 = 5, x_5 = 25, x_6 = 0$ et $Z = 125$.
La fonction économique s'écrit alors :

$$Z = -4x_1 - 2x_2 - x_6 + 125$$

les variables hors-base x_1, x_2 et x_6 sont affectées de coefficients négatifs, Z atteint son maximum au point $A_7(0, 0, 125)$ et vaut 125. La règle du plus grand gain marginal nous a contraint au chemin $OA_1A_2A_3A_4A_5A_6A_7$ de coordonnées respectives $(0, 0, 0), (5, 0, 0), (5, 5, 0), 0, 25, 0), (0, 25, 25), (5, 5, 65)$, $(5, 0, 85), (0, 0, 125)$.
Retour sur le tableau initial :

VHB / VDB	x_1	x_2	x_3	x_4 •	x_5 •	x_6 •	cste	C
x_4	1	0	0	1	0	0	5	$+\infty$
x_5	4	1	0	0	1	0	25	$+\infty$
x_6	8	4	1	0	0	1	125	125
Z	4	2	1	0	0	0	0	

Si on n'utilise pas la règle du plus grand gain marginal et si on décide de faire entrer x_3 en base, x_6 sort de base et le pivot est 1 :

VHB / VDB	x_1	x_2	x_3	x_4	x_5	x_6	cste
x_4	1	0	0	1	0	0	5
x_5	4	1	0	0	1	0	25
x_3	8	4	1	0	0	1	125
Z	-4	-2	0	0	0	-1	-125

les variables hors-base x_1, x_2 et x_6 sont affectées de coefficients négatifs, Z atteint son maximum au point $A_7(0, 0, 125)$ et vaut 125 . Le résultat est cette fois-ci atteint en une seule itération à l'aide de ce qu'on appelle la règle du plus petit gain marginal. Il conviendra de choisir alors parmi les deux règles proposées afin de minimiser les temps de calculs.

2.2.3 Détermination d'une solution de base admissible

Reprenons l'exercice 1 et le cas de l'entreprise Bonvin (1.) mais avec des spécifications supplémentaires : Bonvin s'est engagée à fournir à sa clientèle :

— au moins 15000 hectolitres de vin "Extra",

— et au moins 5000 hectolitres de vin "supérieur".

Sous sa forme canonique le programme linéaire s'écrit :

$$\begin{cases} X_1, X_2 \geq 0 \\ 0,5X_1 + 0,2X_2 \leq 13600 \\ 0,3X_1 + 0,6X_2 \leq 12000 \\ 0,2X_1 + 0,2X_2 \leq 10400 \\ X_1 \leq 20000 \\ X_2 \leq 16000 \\ X_1 \geq 15000 \\ X_2 \geq 5000 \\ \max Z \text{ avec } Z\,(X_1, X_2) = 400X_1 + 500X_2 \end{cases}$$

Les données supplémentaires ont été traduites par les deux dernières contraintes qui sous leur forme standard s'écrivent :

$$X_1 - e_6 = 15000 \quad X_2 - e_7 = 5000 \quad \text{avec } e_6, e_7 \geq 0$$

Dans cette hypothèse, il n'existe plus de base naturelle évidente pour amorcer les calculs car si $X_1 = X_2 = 0$ alors

$$e_6 = -15000 \text{ et } e_7 = -5000$$

ce qui est en contradiction avec les conditions de non-négativité.
Une solution consiste alors à annuler au hasard n variables parmi les $m + n$ variables que comporte le problème (dans le cas de l'exemple précédent, 2 variables parmi 9).
Il y a alors C_{m+n}^n solutions de base envisageables (ici $C_9^2 = 36$). Mais toutes ne sont pas admissibles et, de plus, si le nombre de variables et de contraintes est important, il devient fastidieux de s'en remettre au hasard.
C'est pourquoi une procédure plus méthodique consiste :

1. À introduire dans chaque contrainte h qui pose problème une variable artificielle a_h affectée d'un coefficient égal à 1 .
2. À infliger à chaque variable artificielle une pénalité sous la forme d'un coefficient négatif (dans le cas d'un problème de maximisation) et de valeur absolue très élevée dans la fonction économique originelle.

Ainsi, l'introduction de variables artificielles permet de déterminer simplement une base, certes artificielle, mais admissible pour amorcer l'algorithme.
Les pénalités ont pour objet de provoquer l'élimination des variables artificielles au fil des itérations.
La méthode consiste donc ensuite

1. À retenir comme solution de base initiale la base artificielle telle que :
   - toutes les variables artificielles sont en base (c'est-à-dire non nulles) ;
   - toutes les autres variables des contraintes où figurent des variables artificielles (réelles et d'écart) sont hors base (c'est-à-dire nulles).
2. À appliquer l'algorithme du simplexe jusqu'à ce que toutes les variables artificielles soient supprimées.

Dans le cas étudié, après introduction des variables artificielles a_6 et a_7 respectivement dans les contraintes 6 et 7, le problème s'écrit :

$$\begin{cases} X_1, X_2, e_1, \ldots, e_7, a_6, a_7 \geq 0 \\ 0,5X_1 + 0,2X_2 + e_1 = 13600 \\ 0,3X_1 + 0,6X_2 + e_2 = 12000 \\ 0,2X_1 + 0,2X_2 + e_3 = 10400 \\ X_1 + e_4 = 20000 \\ X_2 + e_5 = 16000 \\ X_1 - e_6 + a_6 = 15000 \\ X_2 - e_7 + a_7 = 5000 \\ \max Z \text{ avec } Z\left(X_1, X_2\right) = 400X_1 + 500X_2 - Ga_6 - Ga_7 \end{cases}$$

Les tableaux ci-dessous montrent qu'après deux itérations, une solution de base admissible est obtenue. Cette base n'est plus artificielle mais réelle. La procédure doit ensuite être poursuivie jusqu'à l'obtention de l'optimum, sans tenir compte des colonnes concernant les variables artificielles.

VHB / VDB	X_1	X_2	e_1 •	e_2 •	e_3 •	e_4 •	e_5 •	e_6	e_7	a_6 •	a_7 •	cste	C
e_1	0,5	0,2	1	0	0	0	0	0	0	0	0	13600	68000
e_2	0,3	0,6	0	1	0	0	0	0	0	0	0	12000	20000
e_3	0,2	0,2	0	0	1	0	0	0	0	0	0	10400	52000
e_4	1	0	0	0	0	1	0	0	0	0	0	20000	$+\infty$
e_5	0	1	0	0	0	0	1	0	0	0	0	16000	16000
a_6	1	0	0	0	0	0	0	-1	0	1	0	15000	$+\infty$
a_7	0	1	0	0	0	0	0	0	-1	0	1	5000	5000
Z	400	500	0	0	0	0	0	0	0	-G	-G	0	

VHB / VDB	X_1	X_2 •	e_1	e_2	e_3	e_4	e_5	e_6	e_7	a_6	cste	C
e_1	0,5	0	1	0	0	0	0	0	0,2	0	12600	25200
e_2	0,3	0	0	1	0	0	0	0	0,6	0	9000	30000
e_3	0,2	0	0	0	1	0	0	0	0,2	0	9400	47000
e_4	1	0	0	0	0	1	0	0	0	0	20000	20000
e_5	0	0	0	0	0	0	1	0	1	0	11000	$+\infty$
a_6	1	0	0	0	0	0	0	-1	0	1	15000	15000
X_2	0	1	0	0	0	0	0	0	-1	0	5000	$+\infty$
Z	400	0	0	0	0	0	0	0	500	-G	-2500000	

VHB / VDB	X_1	X_2	e_1	e_2	e_3	e_4	e_5	e_6	e_7	cste	C
e_1	0	0	1	0	0	0	0	0,5	0,2	5100	
e_2	0	0	0	1	0	0	0	0,3	0,6	4500	
e_3	0	0	0	0	1	0	0	0,2	0,2	6400	
e_4	0	0	0	0	0	1	0	1	0	5000	
e_5	0	0	0	0	0	0	1	0	1	11000	
X_1	1	0	0	0	0	0	0	-1	0	15000	
X_2	0	1	0	0	0	0	0	0	-1	5000	
Z	0	0	0	0	0	0	0	400	500	-8500000	

On peut alors démarrer l'algorithme du simplexe avec la solution admissible $X_1 = 15000$ et $X_2 = 5000$.

1.2.3.2 Utilisation de la méthode du simplexe lorsque la solution optimale n'existe pas

Exemple 1.2.10 *On considère l'exemple suivant :*

$$\begin{cases} x \geq 0, y \geq 0 \\ x - y \leq 30 \\ y - x \leq 40 \\ \quad Maximiser\ Z(x,y) = 2x + 6y. \end{cases}$$

En résolvant graphiquement ce problème on remarque que la solution optimale n'existe pas puisque l'ensemble convexe des solutions réalisables n'est pas borné et la fonction objectif peut augmenter dans ce cas sans limite.
Appliquons l'algorithme du simplexe à cet exemple : la solution $(x, y) = (0, 0)$ est admissible.

VDB	x	y	e_1 •	e_2 •	cste	C
e_1	1	-1	1	0	30	-30
y	-1	1	0	1	40	40
Z	2	6	0	0	0	

VDB	x	y	e_1	e_2	cste	C
e_1	0	0	1	1	70	$+\infty$
y	-1	1	0	1	40	-40
Z	8	0	0	-6	-240	

Aucun coefficient de la colonne sélectionnée n'est positif donc la colonne C ne donne aucune valeur positive non infinie, x peut donc augmenter indéfiniment et la fonction objectif Z également. On dira dans ce cas que la valeur maximale n'existe pas.

1.2.3.3 Utilisation de la méthode du simplexe dans un problème de minimisation

Exemple 1.2.11 *On considère l'exemple suivant :*

$$\begin{cases} x \geq 0, y \geq 0 \\ x - 3y \geq -1 \\ x - y \leq 1 \\ \text{Minimiser } Z(x,y) = -2x + y. \end{cases}$$

On définit dans ce cas une façon de transformer les problèmes de minimisation en problèmes de maximisation. On formule le principe suivant :

$$\text{minimiser } Z = -\text{ maximiser } (-Z)$$

Le problème de minimisation précédent devra donc être transformé en un problème de maximisation soit

$$\begin{cases} x \geq 0, y \geq 0 \\ x - 3y \geq -1 \\ x - y \leq 1 \\ \text{Maximiser } -Z(x,y) = 2x - y \end{cases}$$

Appliquons l'algorithme du simplexe à cet exemple : la solution $(x, y) = (0, 0)$ est admissible.

VHB	x	y	e_1 •	e_2 •	cste	C
e_1	1	-3	-1	0	-1	-1
e_2	1	-1	0	1	1	1
$-Z$	2	-1	0	0	0	

VDB	x •	y	e_1 •	e_2	cste	C
e_1	0	-2	-1	-1	-2	1
x	1	-1	0	1	1	-1
$-Z$	0	1	0	-2	-2	

VDB \ VHB	x	y	e_1	e_2	cste	C
y	0	1	$\frac{1}{2}$	$\frac{1}{2}$	1	
x	1	0	$\frac{1}{2}$	$\frac{3}{2}$	2	
$-Z$	0	0	$-\frac{1}{2}$	$-\frac{5}{2}$	-3	

L'algorithme s'arrête, la solution maximale est $-Z = 3$ si $x = 2$ et $y = 1$. La solution minimale sera donc $Z = -3$ si $x = 2$ et $y = 1$.

1.2.4 Exercices récapitulatifs

Exercice 10. Une entreprise fabrique 3 produits P_1, P_2 et P_3 à partir des 3 composants C_1, C_2 et C_3. Les composants sont acheminés vers l'usine par l'intermédaire d'une société de transport qui facture le coût de transport à l'unité. Les données sont rassemblées dans les tableaux ci-dessous :

	Produits		
	P_1	P_2	P_3
Nombre de composants C_1	1	2	4
Nombre de composants C_2	2	1	2
Nombre de composants C_3	3	2	2

Par exemple, pour fabriquer une unité de produit P_3, il faut 4 composants C_1, 2 composants C_2 et 2 composants C_3.
On se donne ensuite les coûts unitaires transport et hors transport en euros des différents composants :

	C_1	C_2	C_3
Coûts unitaires hors-transport (en euros)	20	25	25
Coûts unitaires transport (en euros)	7	6	5

Les contraintes d'approvisionnement sont telles que l'entrepôt dispose chaque semaine de 70 composants C_1, 80 composants C_2 et 60 composants C_3.
Les marges sur coûts variables unitaires sont de 3 euros pour P_1, 5 euros pour P_2 et 6 euros pour P_3.
On note respectivement x, y et z les nombres d'unités de P_1, P_2 et P_3 fabriquées au cours d'une semaine.

1. Quels sont les coûts totaux hors-transport ainsi que les coûts totaux de transport pour chacun des composants utilisés ?
2. Présenter la forme canonique du programme linéaire permettant de maximiser la marge sur coûts variables hebdomadaires.
3. Présenter la forme standard du programme linéaire permettant de maximiser la marge sur coûts variables hebdomadaires.
4. Déterminer le programme optimal de production. Quelle est la marge correspondante ?
5. Si l'entreprise fabrique le programme optimal, combien reste t -il de composants de chaque sorte ?

Exercice 11. Suite à l'incendie d'un entrepôt, une société fait appel à vos compétences pour reconstituer un programme linéaire retrouvé sur place, dans un état malheureusement assez délabré. Les seules informations dont vous disposez consistent en le tableau donné ci dessous :

VHB / VDB	x	y	z	e_1	e_2	e_3	cste	C
e_1	1	0	0	1	0	0	400	
e_2	2	1	1	0	1	0	1000	
e_3	2	2	3	0	0	1	2000	
Z	20	16	12	0	0	0	0	

1. À l'aide du tableau, déterminer le programme linéaire réalisé par l'entrepôt.
2. Recréer un contexte économique d'entrepôt utilisant les données du tableau précédent.

3. Résoudre le programme linéaire précédent.
4. Comment procède t -on afin de minimiser la fonction objectif ?

Exercice 12. Une entreprise fabrique trois types de piles : sèches de type 1 (PS1), sèches de type 2 (PS2) et à combustible (PC). Le processus de fabrication comporte trois étapes :

— l'assemblage,

— un test de qualité,

— un traitement d'isolation.

Seules les piles satisfaisant le test de qualité sont soumises au traitement d'isolation. Les piles qui ratent le test de qualité sont mises au rebut.
Au cours du mois prochain, l'entreprise disposera en temps-machine de 9000 heures pour l'assemblage, de 1200 heures pour les tests de qualité et de 8500 heures pour le traitement d'isolation. Le tableau suivant résume les informations pertinentes du procédé de fabrication :

Type	Assemblage (seconde/unité)	Test (s/unité)	Isolation (s/unité)	Profit (euros/unités)	Taux d'échec	Perte (euros/unité)
PS1	33	3	15	1,25	3%	0,6
PS2	25	4,5	22	1	1%	0,55
PC	24	4	21	1,1	2%	0,75

1. Quel type de problème reconnaît-on ?
2. Modéliser cet exercice de façon à pouvoir répondre aux questions suivantes :
   (a) Quel est le nombre optimal de piles de chaque type à fabriquer le mois prochain si l'entreprise est assurée de vendre toute sa production ?
   (b) Quel sera le profit ?

Exercice 13. La société SUPERSTOCK désire stocker dans son nouvel entrepôt trois types de produits à savoir des coussins (produit P_1), des traversins (produit P_2) et des couvertures (produit P_3) à l'aide de trois conditionnements particuliers C_1 (sacs), C_2 (palettes) et C_3 (cartons). Les données sont rassemblées dans les tableaux ci-dessous :

	Conditionnement		
	C_1	C_2	C_3
Nombre de coussins P_1	20	40	80
Nombre de traversins P_2	40	20	40
Nombre de couvertures P_3	60	40	40

Les contraintes de fabrication sont telles que l'entreprise stocke journalièrement 1400 coussins, 1600 traversins et 1200 couvertures. Les gains relatifs au stockage et au transport pour les trois conditionnements sont donnés ci-dessous :

	C_1	C_2	C_3
Gains relatifs au stockage (en euros)	46	55	105
Gains relatifs au transport (en euros)	14	45	15

1. Donner la forme canonique du programme linéaire associé.
2. Une résolution graphique du programme linéaire précédent est-elle possible ? Si oui, comment s'y prendre pour trouver la solution optimale ?

3. À l'aide de la méthode du simplexe, résoudre le programme linéaire proposé en utilisant la règle du plus grand gain marginal.
4. Retrouver cette solution à l'aide du plus petit gain marginal.
5. Préciser la valeur maximale de la fonction objectif ainsi que les quantités restantes de coussins, traversins et couvertures.

Exercice 14. Notion de solutions de bases adjacentes.
On considère le problème introductif rappelé ci-dessous :

$$\begin{cases} x_1 \geq 0, x_2 \geq 0 \\ x_1 \leq 4 \\ 2x_2 \leq 12 \leq 18 \\ 3x_1 + 2x_2 \leq 18x_2 \end{cases}$$

1. Écrire le problème sous forme d'égalités en ajoutant les variables d'écart.
2. Considérer toutes les bases possibles en complétant le tableau suivant :

Valeurs variables / VHB	(x_1, x_2)	(x_3, x_4, x_5)	sommet ? (oui/non)
x_1, x_2			
x_1, x_3			
x_1, x_4			
x_1, x_5			
x_2, x_3			
x_2, x_4			
x_2, x_5			
x_3, x_4			
x_3, x_5			
x_4, x_5			

(Pour rappel, une base est obtenue en cherchant l'intersection de deux contraintes prises à l'égalité.)
Vérifier la propriété suivante : "Toute solution de base réalisable correspond à un sommet de la région réalisable".
3. Considérer toutes les solutions de base réalisables. Donner les couples de bases adjacentes en complétant le tableau suivant :

Adjacentes ? (oui/non)	$(0,0)$	$(0,6)$	$(4,0)$	$(4,3)$	$(2,6)$
$(0,0)$	-				
$(0,6)$		-			
$(4,0)$			-		
$(4,3)$				-	
$(2,6)$					-

Exercice 15. Planification de production.
Une compagnie fabrique deux produits dans ses deux ateliers. Les marges unitaires sont respectivement de 2 pour le premier produit et de 1 pour le second. Le temps passé (en heures) dans chacun des ateliers pour fabriquer un produit de chaque type est donné au tableau ci-dessous.

Produit	Produit 1	Produit 2
Atelier	1 h	0 h
Atelier 1	1 h	1 h
Atelier 2		

Les capacités résiduelles sont de 4,5 heures par jour dans l'atelier 1 et de 6 heures par jour dans l'atelier 2 . Les productions non entières sont permises.

1. Formuler mathématiquement le problème.
2. Déterminer la solution optimale au moyen de l'algorithme du simplexe. Préciser, pour chaque itération, la solution de base courante et justifier le choix des variables entrantes et sortantes.
3. Illustrer sur un graphique le chemin suivi par l'algorithme du simplexe.

Chapitre

2

Introduction à la théorie des graphes

2.1 Généralités sur les graphes

Aborder une nouvelle discipline nest pas toujours aisé, car cela implique souvent une succession de définitions. Cest précisément le cas ici. Pour rendre cette introduction plus vivante, nous laccompagnerons, autant que possible, dexemples illustrant lutilité concrète de la théorie des graphes. Comme le suggère le titre de ce chapitre, notre attention se porte sur les graphes, et le lecteur constatera rapidement quils se déclinent en plusieurs formes : graphes simples, multigraphes, digraphes, hypergraphes, etc. Une grande partie de ce chapitre est tirée de [7].

2.1.1 Graphes orientés

Définition 2.1.1 *Soient V un ensemble (fini ou infini) et E une partie de $V \times V$ (i.e., une relation sur V). Le graphe $G = (V, E)$ est la donnée du couple (V, E). Les éléments de V sont appelés les sommets*[1] *ou nœuds de G. Les éléments de E sont appelés les arcs*[2] *ou arêtes de G. Si V est fini, on parlera de graphe fini (en particulier, E est alors fini et contient au plus $(\#V)^2$ arcs).*

Remarque 2.1.2 *Observons que l'ordre au sein des couples appartenant à E est intrinsèquement présent*[3]*. On parlera donc parfois de graphe orienté ou de graphe dirigé. Cette distinction va devenir rapidement indispensable, lorsqu'on introduira les graphes non orientés.*

Soit I, un ensemble d'indices. Si $V = \{v_i \mid i \in I\}$ et si $a = (v_i, v_j)\,, i, j \in I$, on pourra alors parler de l'origine v_i et de la destination v_j de l'arc a. On dit que v_i et v_j sont les extrémités de l'arc a et que a relie v_i à v_j. Si $b = (v_i, v_i)$, on parle généralement de la boucle b. Il est souvent commode de donner une représentation sagittale d'un graphe. Les sommets sont représentés par des points et si (v_i, v_j)

1. En anglais, cela se dit *vertex* (au pluriel, *vertices*). D'où l'initiale V pour désigner l'ensemble des sommets. Dans ces notes, nous ne dérogerons pas à la coutume anglosaxonne de noter un graphe $G = (V, E)$.
2. En anglais, cela se dit *edge*. D'où l'initiale usuelle E.
3. On parle de couple et non de paire. Un couple est une paire ordonnée. On distingue d'ailleurs les notations (x, y) et $\{x, y\}$.

est un arc, alors on trace une flèche de v_i vers v_j (cf. figure 2.1). Deux arcs sont *adjacents* s'ils ont au moins une extrémité en commun.

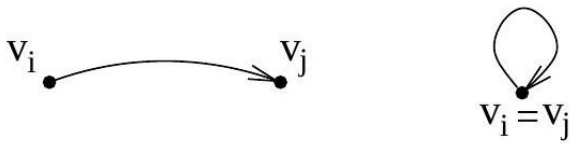


FIGURE 2.1 – Un arc reliant deux sommets, une boucle.

Définition 2.1.3 *Soit $a = (v_i, v_j) \in E$. On dit que a est un arc sortant de v_i ou encore que a est un arc incident à v_i vers l'extérieur (resp. un arc entrant dans v_j ou encore que a est un arc incident à v_j vers l'intérieur).*
L'ensemble des arcs sortant de v_i est noté $\omega^+(v_i)$ et l'ensemble des arcs entrant dans v_j est noté $\omega^-(v_j)$. L'ensemble des arcs incidents à un sommet v est $\omega(v) := \omega^+(v) \cup \omega^-(v)$. On définit le demi-degré sortant (resp. demi-degré entrant) d'un sommet v par

$$d^+(v) = \#\left(\omega^+(v)\right) \quad \left(\text{ resp.} d^-(v) = \#\left(\omega^-(v)\right)\right).$$

Si $G = (V, E)$ est un graphe fini, il est clair que

$$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v). \tag{2.1}$$

Enfin, le degré de v est $\deg(v) = d^+(v) + d^-(v)$. L'ensemble des successeurs d'un sommet v est l'ensemble $\text{succ}(v) = \{s_1, \ldots, s_k\}$ des sommets s_i tels que $(v, s_i) \in \omega^+(v)$, i.e., $(v, s_i) \in E$. De manière analogue, l'ensemble des prédécesseurs d'un sommet v est l'ensemble $\text{pred}\,(v) = \{s_1, \ldots, s_k\}$ des sommets s_i tels que $(s_i, v) \in \omega^-(v)$, i.e., $(s_i, v) \in E$. Enfin, l'ensemble des voisins de v est simplement

$$\nu(v) = \text{pred}(v) \cup \text{succ}(v).$$

Si u appartient à $\nu(v)$, on dit que u et v sont des sommets *voisins* ou *adjacents.*

Exemple 2.1.4 *Soit le graphe $G = (V, E)$ où $V = \{a, b, c, d, e\}$ et*

$$E = \{(a, b), (a, e), (b, b), (b, c), (c, c), (c, d), (c, e), (d, a), (e, a), (e, d)\}$$

Celui-ci est représenté à la figure 2.2. Par exemple, $\omega^+(a) = \{(a, b), (a, e)\}$ et $\omega^-(d) = \{(c, d), (e, d)\}$.

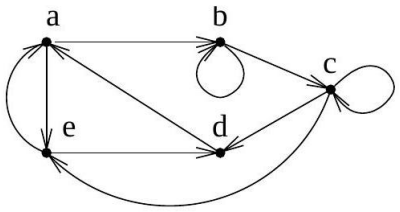


FIGURE 2.2 – Un exemple de graphe.

On a aussi $\text{succ}(a) = \{b, e\}, \text{succ}(b) = \{b, c\}$, $\text{pred}(d) = \{c, e\}$ et $\nu(a) = \{b, d, e\}$. On voit aussi que les arcs (e, a) et (d, a) sont adjacents. Enfin, le demi-degré sortant de c est $d^+(c) = 3$.

Définition 2.1.5 *Un multi-ensemble*[4] *est un ensemble au sein duquel un même élément peut être répété plus d'une fois. Ainsi, on s'intéresse non seulement à savoir si un élément appartient ou non à un multi-ensemble donné, mais également à sa multiplicité. Par exemple,* $\{1,1,2,3\}, \{1,2,3\}$ *et* $\{1,2,2,3\}$ *sont des multi-ensembles distincts.*

Pour distinguer les copies d'un même élément x, il est commode de les indicer. Par exemple, on considère le multi-ensemble $\{1_1, 1_2, 1_3, 2_1, 2_2, 3\}$. Cette manière de procéder nous permettra de définir facilement des fonctions définies sur un multi-ensemble.
Un multi-graphe $G = (V, E)$ est un graphe pour lequel l'ensemble E des arcs est un multi-ensemble. Autrement dit, il peut exister plus d'un arc reliant deux sommets donnés. Un exemple de représentation d'un multigraphe est donné à la figure 2.3 Un multi-graphe $G = (V, E)$ est fini si V. et E sont

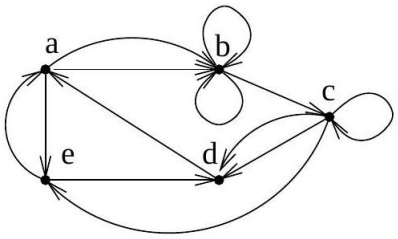


FIGURE 2.3 – Un exemple de multi-graphe.

finis. (En effet, dans le cas des multi-graphes, supposer V fini n'implique pas que E soit fini.)
Soit $p \geq 1$. Un p-graphe est un multi-graphe $G = (V, E)$ pour lequel tout arc de E est répété au plus p fois. En particulier, un 1-graphe est un graphe.

Remarque 2.1.6 *On peut observer que la remarque 2.1.2, la définition 2.1.3 et la "handshaking formula" s'appliquent également au cas des multigraphes. Il est laissé au lecteur le soin d'adapter les définitions de* $\omega^+(v)$, $d^+(v), \text{succ}(v)$ *et* $\omega^-(v), d^-(v), \text{pred}(v)$. *En particulier,* $\omega^+(v)$ *et* $\omega^-(v)$ *sont en général des multi-ensembles.*

Définition 2.1.7 *Un graphe* $G = (V, E)$ *est dit simple (ou strict) s'il ne s'agit pas d'un multi-graphe et si* E *est irréflexif, c'est-à-dire que quel que soit* $v \in V$, (v, v) *n'appartient pas à* E *(i.e.,* G *ne contient pas de boucle). Un exemple de graphe simple est donné à la figure 2.4.*

On suppose qu'un élément est répété au plus un nombre dénombrable de fois.

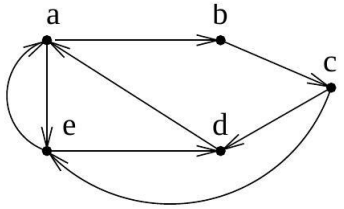


FIGURE 2.4 – Un exemple de graphe simple.

4. Cette définition n'est pas très rigoureuse. En effet, le concept même d'ensemble ne vous a, jusqu'à présent, été introduit que de manière naïve.

2.1.2 Graphes non orientés

Les graphes non orientés sont en fait un cas particulier de graphes (orientés).

Définition 2.1.8 *Soit $G = (V, E)$ un graphe (resp. un multi-graphe). Si E est une relation symétrique sur V, on dira que G est un graphe (resp. un multi-graphe) non dirigé ou non orienté. Autrement dit, G est non dirigé si*

$$\forall v_1, v_2 \in V : (v_1, v_2) \in E \Rightarrow (v_2, v_1) \in E.$$

Dans ce cas, on simplifie la représentation sagittale de G en traçant simplement un segment entre v_1 et v_2. Pour alléger l'écriture, on identifiera les arcs (v_i, v_j) et (v_i, v_j) avec une unique "arête non orientée" donnée par la paire $\{v_i, v_j\}$. Dans le cas dirigé (resp. non dirigé), nous nous efforcerons de parler d'arcs (resp. d'arêtes).

Si par contre, on désire insister sur le caractère non symétrique de E, on parlera de graphe dirigé ou, par abus de langage, digraphe. Les définitions rencontrées précédemment s'adaptent aisément au cas non orienté.

Définition 2.1.9 *Soient $G = (V, E)$, un multi-graphe non orienté et $a = \{v_i, v_j\}$ une de ses arêtes. On dit que a est incident aux sommets v_i et v_j. Le nombre d'arêtes incidentes à v_i est le degré de v_i, noté $\deg(v_i)$. On suppose en outre que les boucles apportent une double contribution au degré d'un sommet. L'ensemble des arêtes incidentes à v_i se note $\omega(v_i)$. Il est clair que, dans un graphe simple, $\deg(v_i) = \#(\omega(v_i))$. Ces notations sont bien évidemment compatibles avec celles données dans le cas orienté. Deux arêtes sont adjacentes si elles ont au moins une extrémité en commun.*

Deux sommets $v_i, v_j \in V$ sont adjacents si l'arête $\{v_i, v_j\}$ appartient à E. On dit aussi qu'ils sont voisins. L'ensemble des voisins de v se note $\nu(v)$. Enfin, la définition d'un p-graphe est analogue à celle donnée dans le cas orienté.

Lemme 2.1.10 *(Handshaking lemma). Si $G = (V, E)$ est un multigraphe non orienté, alors*

$$\sum_{v \in V} \deg(v) = 2\#E. \tag{2.2}$$

Démonstration. C'est immédiat. (Et on comprend mieux la double contribution des boucles pour le degré d'un sommet....)
□

C'est le lemme dit de *des poignées de main.* Il signifie que, par exemple lors d'un festin, *la somme des poignées de main individuelles = 2 × Poignées de main totales.*
L'exemple suivant illustre les différentes classes de graphes rencontrées jusqu'à présent. Bien sûr, tout graphe simple est un graphe et tout graphe est un multi-graphe.

Exemple 2.1.11 *A la figure 2.5, on a représenté, dans le cas dirigé, un graphe simple, un graphe et enfin, un multi-graphe. La figure 2.6 reprend les mêmes éléments dans le cas non orienté.*

Définition 2.1.12 *Soit $k \geq 1$. Un multi-graphe orienté (resp. non orienté) $G = (V, E)$ est k-régulier si pour tout $v \in V, d^+(v) = k$ (resp. $\deg(v) = k$). Le graphe de gauche (resp. de droite) de la figure 2.7 est 3-régulier (resp. 4-régulier). Le graphe de droite de la figure 2.7 est en particulier simple et complet.*

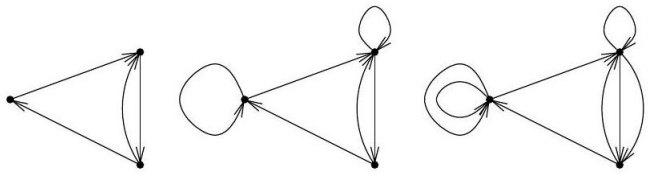


FIGURE 2.5 – Un graphe (dirigé) simple, un graphe et un multi-graphe.

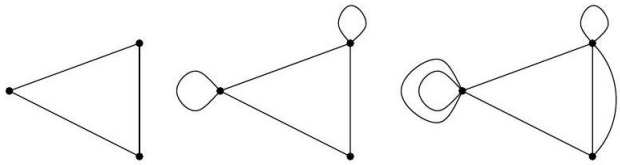


FIGURE 2.6 – Un graphe (non dirigé) simple, un graphe et un multi-graphe.

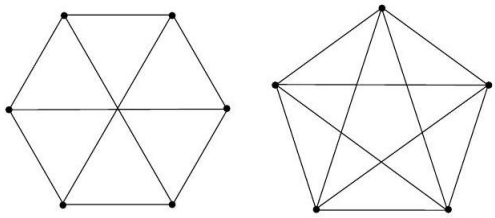


FIGURE 2.7 – Des graphes non orientés 3 et 4-réguliers.

Un graphe $G = (V, E)$ est complet si $E = V \times V$, plus exactement, on suppose souvent que

$$E = V \times V \backslash \{(v, v) \mid v \in V\}$$

(autrement dit, on ne tient pas compte des boucles). En particulier, un graphe complet est symétrique. On note K_n le graphe simple non orienté complet à n sommets. Ainsi, la figure 2.7 représente le graphe K_5. Dans ce cours, lorsqu'on parlera de graphes complets, il sera sous-entendu qu'il s'agit de graphes simples et non orientés.

Définition 2.1.13 *Un graphe $G = (V, E)$ est dit biparti si V peut être partitionné en deux ensembles V_1 et V_2 de manière telle que $E \subseteq V_1 \times V_2$. Si $\#V_1 = m, \#V_2 = n$ et $E = V_1 \times V_2$, alors on parle du graphe biparti complet et il est noté $K_{m,n}$. On peut généraliser cette notion et définir des graphes n-partis, pour $n \geq 2$. Pour ce faire, V doit être partitionné en n sous-ensembles $V_1, \ldots, V_n$ de manière telle que*

$$E \subseteq \bigcup_{i \neq j} V_i \times V_j.$$

Définition 2.1.14 *Un multi-graphe $G = (V, E)$ (orienté ou non) est étiqueté (par f) s'il existe une fonction*

$$f : E \to \Sigma$$

où Σ est un ensemble quelconque. Si $\Sigma \subseteq \mathbb{R}^+ = [0, +\infty[$, on parle souvent de multi-graphe pondéré et on dit que f est une fonction de poids. Un étiquetage peut par exemple servir à préciser des coûts

4. En anglais *directed graph* se contracte en *digraph*.

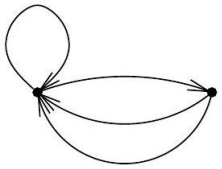


FIGURE 2.8 – Un multi-graphe orienté 2-régulier.

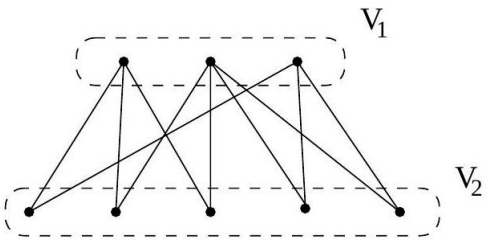


FIGURE 2.9 – Un graphe biparti (non complet).

(coût de transport, des distances, des couleurs, etc...). Si a est un arc, $f(a)$ est l'étiquette, le label ou encore le poids de a. On peut de la même manière définir un étiquetage des sommets au moyen d'une fonction $g : V \to \Sigma$.

Exemple 2.1.15 *Le graphe de la figure 2.10 représente quelques villes belges connectées par un réseau autoroutier. L'étiquette de chaque arête représente la distance, par autoroute, entre les deux extrémités de celle-ci. Nous avons choisi un graphe non orienté car les autoroutes belges vont toujours dans les deux sens.*

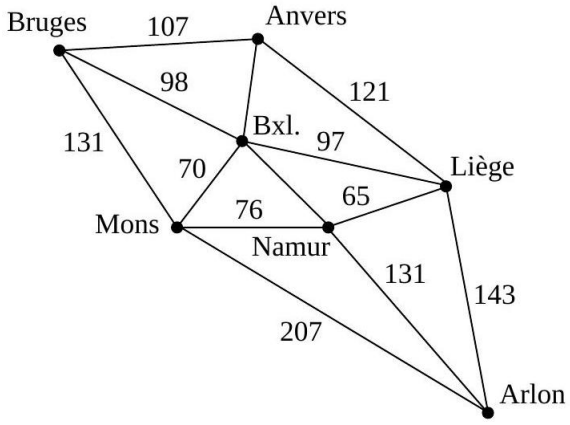


FIGURE 2.10 – Graphe étiqueté par les distances entre villes (itinéraire "express" calculé par www.mappy.fr).

Pour terminer cette section, nous généralisons le concept de graphe en autorisant non plus des relations binaires entre sommets (autrement dit, des arcs) mais des relations d'arité quelconque (des hyper-arêtes). Ensuite, nous donnons une brève introduction aux matroïdes.

Définition 2.1.16 *Un hyper-graphe $H = (V, E)$ est la donnée de deux ensembles V et E. L'ensemble V, comme dans le cas d'un graphe, est l'ensemble des sommets de H. Par contre, E est une partie de*

$\mathcal{P}(V)$ (l'ensemble des parties de V). Un élément de E est appelé hyper-arête. Soient $V = \{a, b, c, d, e\}$ et

$$E = \{\{a, b\}, \{a, c, d\}, \{b, d, e\}\}.$$

On dispose de l'hyper-graphe $H = (V, E)$ représenté à la figure 2.11. Un hyper-graphe $H = (V, E)$ est fini si V est fini. La notion de matroïde est dûe à H. Whitney (1935) et a été développée par

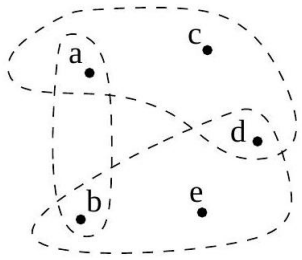


FIGURE 2.11 – Un exemple d'hyper-graphe.

W. Tutte. Elle permet notamment l'étude axiomatique des propriétés de l'indépendance linéaire ou aussi l'étude des cycles et des arbres.

Définition 2.1.17 *Un matroïde $(M, \mathcal{I})$ est la donnée d'un ensemble fini M et d'une partie $\mathcal{I}$ de $\mathcal{P}(M)$, i.e., d'une collection de sous-ensembles de M, vérifiant les trois propriétés suivantes*

- *$\emptyset \in \mathcal{I}$ (de manière équvalente, $\mathcal{I}$ est non vide),*
- *si $X \in \mathcal{I}$ et $Y \subseteq X$, alors $Y \in \mathcal{I}$,*
- *si $X, Y \in \mathcal{I}$ et $\#X > \#Y$, alors il existe $x \in X \backslash Y$ tel que $Y \cup \{x\} \in \mathcal{I}$.*
  Les ensembles de $\mathcal{I}$ sont dits indépendants. Enfin, un matroïde est qualifié de normal si tous les singletons sont indépendants.

Exemple 2.1.18 *Soient E un espace vectoriel, M une partie finie de E et $\mathcal{I}$ la collection des parties libres de M. Le couple $(M, \mathcal{I})$ est un matroïde.*

Exemple 2.1.19 *Soit $G = (V, E)$ un graphe non orienté simple. On considère le matroïde $(E, \mathcal{I})$ où une partie X de E appartient à $\mathcal{I}$ si et seulement si X ne contient aucun cycle (i.e., X est une forêt).*

Remarque 2.1.20 *Tout matroïde $(M, \mathcal{I})$ est bien évidemment un hypergraphe. Les sommets de ce dernier sont les éléments de M et les hyper-arêtes sont les éléments de $\mathcal{I}$.*

2.1.3 Quelques illustrations typiques des graphes

Nous allons présenter dans cette courte section quelques exemples de graphes permettant la modélisation de divers problèmes. Puisqu'il ne s'agit que d'exemples, certaines définitions sont volontairement omises. Elles seront précisées en temps utiles. Nous espérons que la variété des exemples présentés servira de motivation profonde à l'étude théorique réalisée dans les sections et chapitres suivants. Pour des raisons évidentes de présentation, nous ne donnons que des exemples de "petite taille" qui peuvent souvent être résolus sans véritable méthode. Dans des problèmes réels, il faut imaginer des graphes pouvant avoir plus de 10^6 sommets. Dans ce cas, la solution paraît nettement moins évidente !

Exemple 2.1.21 *(Circuit eulérien). Les ouvrages de théorie des graphes reprennent toujours le célèbre exemple des ponts de Königsberg. Nous ne dérogerons pas à cette règle. Au XVII-ième siècle, les habitants de Königsberg (actuel Kaliningrad, ville de Russie proche de la Lituanie et de la Pologne où coule la rivière Pregel) désiraient se promener le dimanche en passant une et une seule fois par chacun des sept ponts de la ville. Les ponts étaient disposés comme à la figure 2.12. Une modélisation du problème revient à considérer un graphe ayant comme sommets, les deux rives et les deux îles et comme arêtes, les sept ponts. Puisqu'il n'y a aucune contrainte sur le sens de parcours des ponts, nous avons choisi un multi-graphe non orienté.*

La question générale sous-jacente est donc de déterminer pour un multigraphe donné (éventuellement orienté) s'il existe un circuit, i.e., un chemin fermé, passant une et une seule fois par chaque arête.

Exemple 2.1.22 *(Circuit hamiltonien - TSP). Le problème du voyageur de commerce (Travel Salesman Problem) est en quelque sorte un "problème dual" de l'exemple précédent (on s'intéresse ici à trouver un circuit passant une et une seule fois par chaque sommet et non par chaque arête). On cherche un circuit permettant non seulement de relier n villes en passant une et une seule fois par chacune d'entre elles, mais de plus, ce circuit doit minimiser la distance totale parcourue. On pourrait par exemple considérer les villes belges et les données de la figure* **??** *et en déterminer un circuit optimal. Dans certains cas, on a recours à un graphe orienté plutôt qu'à un graphe non orienté comme à la figure 2.10. Cela a pour avantage de permettre la modélisation de coûts différents pour aller d'un sommet A à un sommet B plutôt que de B à A (ceci permet, par exemple, de prendre en compte des sens uniques, des payages, des temps de transports différents suivant la direction choisie, etc...). Il s'agit donc de problèmes typiques rencontrés dans la détermination de tournées, de circuits de distribution, etc...*

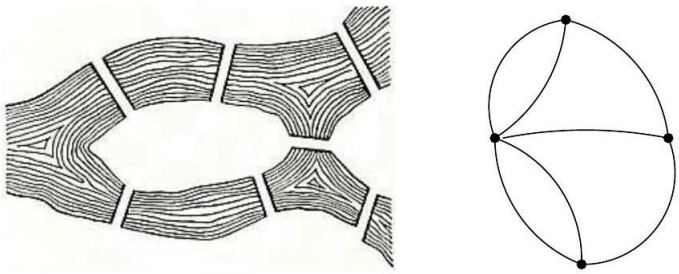


FIGURE 2.12 – Les sept ponts de Königsberg.

Exemple 2.1.23 *(Coloriage). Considérons un cube. Si chaque sommet (resp. chaque arête) d'un graphe représente un sommet (resp. une arête) du cube, on obtient le graphe de gauche de la figure 2.13, on parle du squelette du cube. Par contre, si on représente les faces du cube par les sommets d'un graphe et si les sommets du graphe correspondant à des faces du cube ayant une arête commune sont adjacents, on obtient celui de droite. On peut alors poser la question générale de déterminer le nombre de couleurs nécessaire et suffisant pour colorier les sommets d'un multi-graphe donné, de manière telle que deux sommets adjacents ne reçoivent pas la même couleur. Si on répond à cette question dans le cas de notre exemple initial, on s'aperçoit que pour colorier les sommets (resp. les faces d'un cube), deux (resp. trois) couleurs sont suffisantes.*

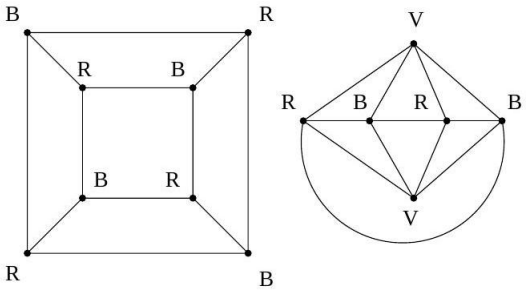


FIGURE 2.13 – Deux représentations d'un cube.

Exemple 2.1.24 *(Cartographie). Quel est le nombre minimum de couleurs nécessaires pour colorier les pays d'une carte géographique de manière telle que des couleurs distinctes soient attribuées à des pays voisins ? Ce problème se ramène au précédent. On considère un graphe ayant pour sommet les*

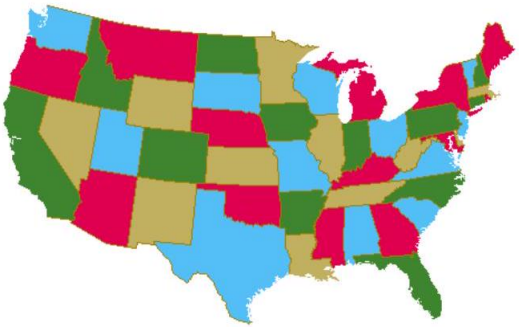


FIGURE 2.14 – La carte des USA.

différents pays de la carte. Deux sommets sont adjacents si et seulement si ils partagent une frontière.

Exemple 2.1.25 *(Graphe d'incompatibilité). Pour le transport de produits chimiques par rail, certains de ces produits ne peuvent être transportés dans un même wagon (des produits co-combustibles sont placés dans des wagons distincts). La figure 2.15 représente le graphe d'incompatibilité de transport (deux sommets adjacents ne peuvent être placés dans le même wagon). On demande de minimiser le nombre de wagons nécessaires au transport. Par rapport à l'exemple précédent, le graphe n'est pas nécessairement planaire. problème se ramène donc à un problème de coloriage. Deux sommets adjacents doivent avoir des couleurs distinctes, une couleur correspondant à un wagon.*

Exemple 2.1.26 *(Coloriage d'arêtes). Au lieu de vouloir colorier les sommets d'un multi-graphe, on peut aussi s'intéresser au problème suivant. Déterminer le nombre de couleurs nécessaires et suffisantes pour colorier les arêtes d'un multi-graphe donné, de manière telle que deux arêtes adjacentes ne reçoivent pas la même couleur.*

Exemple 2.1.27 *(Graphe de Cayley). Soit G un groupe et S un ensemble de générateurs de G (tout élément de G s'obtient comme produit d'un nombre fini d'éléments de S ou d'inverses d'éléments de S). Le graphe de Cayley du groupe G (par rapport à S) est un graphe orienté $\mathcal{C}_S(G)$ ayant pour*

4. La figure 2.14 est tirée de http ://www.math.gatech.edu/ ~ thomas/FC/fourcolor.html

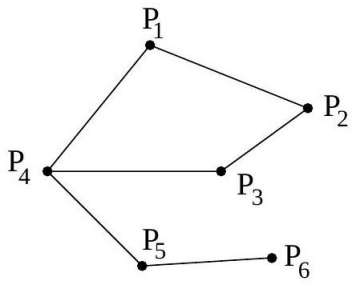


FIGURE 2.15 – Graphe d'incompatibilité.

sommets les éléments de G. Ses arcs sont définis comme suit. Pour tous $g \in G, s \in S$, l'arc (g, gs) est un arc du graphe ayant s pour label. Soit le groupe symétrique $\mathcal{S}_3$ des permutations à 3 éléments ayant pour ensemble générateur

$$S = \left\{x = \begin{pmatrix} 1 & 2 \end{pmatrix}, y = \begin{pmatrix} 1 & 3 \end{pmatrix}, z = \begin{pmatrix} 2 & 3 \end{pmatrix}\right\}$$

Il est clair que

$$xx = yy = zz = id, \quad xy = \begin{pmatrix} 1 & 2 \end{pmatrix}\begin{pmatrix} 1 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 2 \end{pmatrix},$$
$$xz = \begin{pmatrix} 1 & 2 \end{pmatrix}\begin{pmatrix} 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}, \quad yz = \begin{pmatrix} 1 & 3 \end{pmatrix}\begin{pmatrix} 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 2 \end{pmatrix}$$

et

$$yx = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}, zx = \begin{pmatrix} 1 & 3 & 2 \end{pmatrix}, zy = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}.$$

Le graphe de Cayley correspondant est représenté à la figure 2.16.

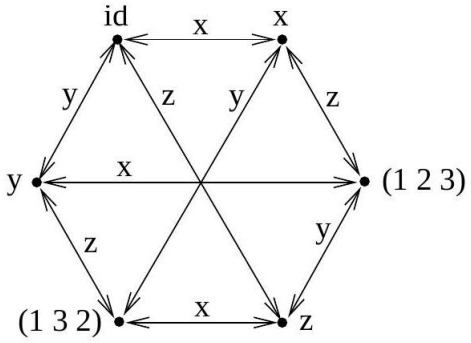


FIGURE 2.16 – Graphe de Cayley de $\mathcal{S}_3$.

On peut obtenir un autre graphe de Cayley de $\mathcal{S}_3$ en considérant comme ensemble de générateurs $\left\{a = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}, b = \begin{pmatrix} 1 & 2 \end{pmatrix}\right\}$. On obtient alors le graphe de la figure 2.17 (les détails de calcul sont laissés au lecteur). Bien évidemment, le graphe de Cayley d'un groupe ne donne aucun renseignement qui ne saurait être obtenu sous une forme "algébrique" classique. Néanmoins, un tel diagramme a l'avantage de fournir presque immédiatement certaines informations qui seraient plus pénibles à obtenir par d'autres moyens (imaginez un groupe défini par certaines relations, par exemple, $cd = dc, c^2dc = d$, etc. . .). Ainsi, il est ici immédiat de vérifier sur la figure 2.17 que les éléments baaba et a^2 sont identiques (il suffit de suivre les labels de chemins depuis id). L'utilisation des graphes de

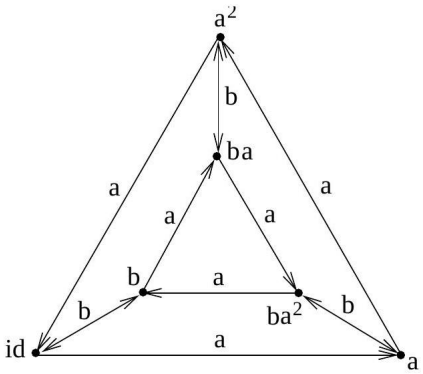


FIGURE 2.17 – Un autre graphe de Cayley de $\mathcal{S}_3$.

Cayley se révèle être un outil souvent bien utile en théorie des représentations ou dans la résolution de problèmes précis à l'aide de l'ordinateur.

2.2 Chemins et circuits

Les définitions suivantes sont somme toute assez naturelles et intuitives, mais il faut bien les préciser au moins une fois de manière rigoureuse pour savoir de quoi on parle exactement.

Définition 2.2.1 *Soit $G = (V, E)$ un multi-graphe non orienté*
Un chemin de longueur $k \geq 1$ est une suite ordonnée $(e_1, \ldots, e_k)$ de k arêtes adjacentes $e_i = \{e_{i,1}, e_{i,2}\}$, i.e., pour tous $i \in \{1, \ldots, k-1\}, e_{i,2} = e_{i+1,1}$. Ce chemin de longueur k joint les sommets $e_{1,1}$ et $e_{k,2}$. On dit que le chemin $(e_1, \ldots, e_k)$ passe par les arêtes $e_1, \ldots, e_k$ (resp. par les sommets $e_{1,1}, e_{2,1}, \ldots, e_{k,1}$ et $e_{k,2}$). On supposera qu'un chemin de longueur 0 (correspondant à la suite vide) joint toujours un sommet à lui-même.

Si les extrémités du chemin sont égales, i.e., si $e_{1,1} = e_{k,2}$, on parle plutôt de cycle, de circuit ou encore de chemin fermé. Si on désire préciser que le chemin considéré n'est pas un cycle, on parlera de chemin ouvert.
Il se peut que les arêtes d'un chemin soient toutes distinctes (cela n'implique pas que les sommets du chemin soient tous distincts). On parle alors de piste ou de chemin élémentaire (voir par exemple, la figure 2.18). Si les arêtes d'un chemin sont toutes distinctes et si de plus, les sommets sont tous distincts [5], on parle alors de chemin simple. Bien sûr, les circuits étant des chemins particuliers, on parle aussi de circuit élémentaire ou simple (voir par exemple, la figure 2.19). Evidemment, dans la définition d'un circuit simple, on admet que les sommets $e_{1,1}$ de départ et $e_{k,2}$ d'arrivée puissent être égaux, mais seulement eux. Il est clair que tout circuit peut se décomposer en circuits simples.

Remarque 2.2.2 *Suivant les auteurs, la terminologie peut changer énormément. Ainsi, il n'est pas rare (par exemple, pour C. Berge), d'inverser les définitions des adjectifs simple et élémentaire. Pour d'autres, la notion de cycle contient de plus le caractère élémentaire, voire simple.*

4. Ce graphe peut avoir ou non des boucles, peu importe.
5. Sauf dans le cas pathologique du circuit $(\{a, b\}, \{b, a\})$, demander que les sommets d'un chemin soient tous distincts (mis à part les extrémités du chemin dans le cas d'un circuit), implique que les arêtes sont aussi toutes distinctes.

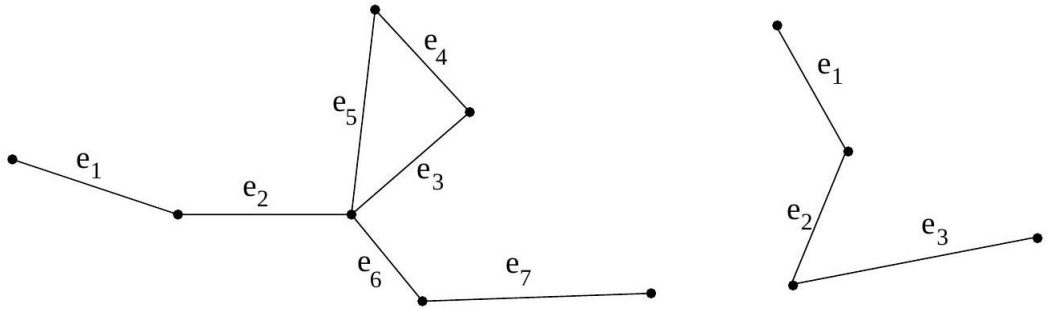


FIGURE 2.18 – Une piste (ou chemin élémentaire) et un chemin simple

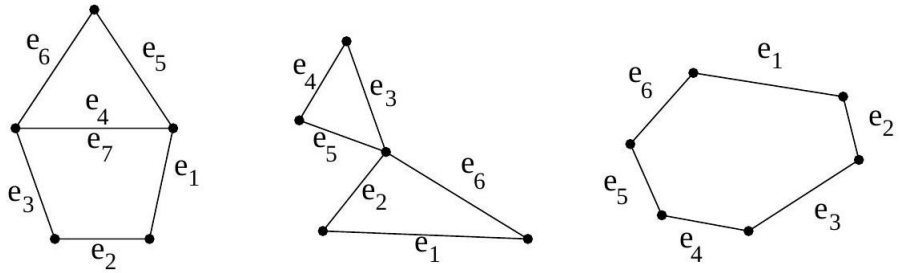


FIGURE 2.19 – Un circuit, un circuit élémentaire (ou piste fermée) et un circuit simple.

Remarque 2.2.3 *Dans le cas d'un graphe (qui n'est pas un multi-graphe), c'est-à-dire pour un 1-graphe, un chemin est aussi univoquement déterminé par une suite de sommets* $(v_1, \ldots, v_k)$ *de manière telle que* $\{v_i, v_{i+1}\}$ *est une arête du graphe.*

Définition 2.2.4 *Deux sommets* a *et* b *sont connectés s'il existe un chemin les joignant, ce que l'on notera* $a \sim b$*. La relation* $\sim$ *"être connecté" est une relation d'équivalence sur* V*. Une classe d'équivalence pour* $\sim$ *est une composante connexe de* G*. Un multi-graphe non orienté est connexe si* $V/\sim$ *contient une seule classe d'équivalence, i.e.,* G *possède une seule composante connexe. Autrement dit, un multi-graphe non orienté est connexe si, pour toute paire de sommets, il existe un chemin les joignant. On supposera de plus que* $G = (\{v\}, \emptyset)$ *est connexe (ce qui revient à supposer qu'un chemin de longueur 0 joint toujours un sommet à lui-même).*

Définition 2.2.5 *Soit* $G = (V, E)$ *un multi-graphe non orienté connexe. La distance*[6] *entre deux sommets* a *et* b *est la longueur du plus court chemin joignant* a *et* b*. On la note* $\mathrm{d}(a, b)$*. Le diamètre de* G *est défini par*

$$\mathrm{diam}(G) = \max_{a,b \in V} \mathrm{d}(a, b).$$

Si G est en outre pondéré par la fonction de poids $f : E \to \mathbb{R}^+$, la distance entre les sommets a et b est égale au poids minimal des chemins joignant a et b, i.e.,

$$\mathrm{d}(a, b) = \min_{\substack{\text{chemin } (e_1,\ldots,e_t) \\ \text{joignant } a \text{ et } b}} \sum_{i=1}^{t} f(e_i).$$

5. Si G n'était pas connexe, la fonction distance ne serait pas une fonction totale définie sur $V \times V$ tout entier.
6. Vérifier qu'il s'agit effectivement d'une distance.

Les définitions données précédemment s'adaptent aisément au cas d'un multi-graphe orienté. Il suffit de respecter en plus le sens de parcours imposé par les arcs. Donnons quelques précisions.

Définition 2.2.6 *Soit $G = (V, E)$ un multi-graphe orienté*[7]*. Un chemin de longueur $k \geq 1$ est une suite ordonnée $(v_1, \ldots, v_k)$ de k arcs $v_i = (v_{i,1}, v_{i,2})$ tels que pour tous $i \in \{1, \ldots, k-1\}, v_{i,2} = v_{i+1,1}$. Ce chemin de longueur k joint les sommets $v_{1,1}$ et $v_{k,2}$.*

S'il existe un chemin joignant deux sommets a et b, on notera $a \to b$. Si $a \to b$ et $b \to a$, on dira que a et b sont fortement connectés et on notera $a \leftrightarrow b$. Si on impose à $\leftrightarrow$ d'être réflexive (i.e., on suppose que $a \leftrightarrow a$), on vérifie aisément que la relation $\leftrightarrow$ "être fortement connecté" est une relation d'équivalence sur V. Une classe d'équivalence pour $\leftrightarrow$ est une composante fortement connexe (ou, plus court, $f.$ connexe) de G. Si $V/\leftrightarrow$ contient une seule classe, on dira que G est fortement connexe (ou $f.$ connexe).
Les sommets appartenant à un cycle maximal, i.e., un cycle auquel on ne peut adjoindre de nouveaux sommets, constituent une composante $f.$ connexe. Autrement dit, un multi-graphe orienté G est f. connexe si et seulement si il existe un cycle passant par chaque sommet de celui-ci.
Si on supprime l'orientation des arcs de G et si le multi-graphe non orienté obtenu de cette manière est connexe, alors on dira que G est simplement connexe (ou s. connexe). On pourra bien entendu définir, de manière évidente, les composantes simplement connexes (ou s. connexe de G).

Remarque 2.2.7 *Les notions de distance et de diamètre données dans le cas non orienté s'adaptent facilement au cas d'un multi-graphe orienté fortement connexe. On remarquera cependant qu'ici, la fonction $\mathrm{d}(\cdot, \cdot)$ n'est en général pas symétrique*[8]*.*

Exemple 2.2.8 *Considérons le multi-graphe orienté de la figure 2.20 dont l'ensemble des sommets est $\{a, \ldots, e\}$ et l'ensemble des arcs est $\{1, \ldots, 7\}$. Ce graphe est simplement connexe mais il n'est pas fortement connexe. En effet, il n'existe pas de chemin joignant les sommets d à a. L'ensemble $\{b, c, d\}$ est une composante fortement connexe du graphe (les deux autres composantes sont $\{a\}$ et $\{e\}$). Un chemin est par exemple donné par $(1, 3, 7)$, un cycle par $(3, 4, 5)$ et une piste par $(1, 3, 4, 5, 6)$. La distance entre d et c vaut 1. Par contre, la distance entre c et d vaut 2.*

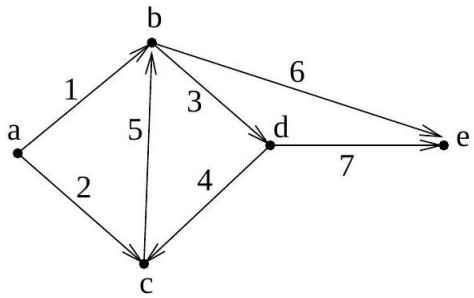


FIGURE 2.20 – Un graphe orienté.

7. Ce graphe peut avoir ou non des boucles, peu importe.
8. Il ne peut donc pas s'agir à proprement parler d'une distance.

2.2.1 Recherche du plus court chemin : Algorithme de Dijkstra

Soit $G = (V, E)$ un digraphe pondéré par la fonction $p : E \to \mathbb{R}^+$. Nous allons présenter l'algorithme de Dijkstra de recherche d'un plus court chemin (i.e., un chemin de poids minimal) d'un sommet u fixé à un sommet quelconque de G. Il est clair que l'on peut restreindre ce problème au cas d'un digraphe simple. Pour rappel, si u et v sont deux sommets tels que $u \to v$, le poids d'un chemin $(e_1, \ldots, e_k)$ les joignant est $\sum_i p(e_i)$. La distance de u à v est alors le poids minimal de tels chemins. Si $u \nrightarrow v$, on n'a pas à proprement parler de distance [9] et par convention, on posera que la "distance" de u à v est alors $+\infty$.
L'algorithme de Dijkstra s'applique également à un graphe non orienté. Il suffit de remplacer l'arête $\{u, v\}$ de poids α par deux arcs (u, v) et (v, u) ayant chacun un poids α et ainsi obtenir un digraphe.

Remarque 2.2.9 *Quitte à supposer p à valeurs dans $\mathbb{R}^+ \cup \{+\infty\}$, on étend p de E à $V \times V$ tout entier en posant $p(x, x) = 0$, pour tout $x \in V$ et $p(x, y) = +\infty$, si $(x, y) \notin E$. C'est ce que nous supposerons par la suite.*

Intuitivement, l'algorithme fonctionne de la manière suivante. A chaque sommet v de G, on associe une valeur $\mathrm{T}(v)$, initialisée à $p(u, v)$ et une liste de sommets $\mathrm{C}(v)$ censée correspondre à un chemin de u à v. Lorsque l'algorithme s'achève, $\mathrm{T}(v)$ contient le poids minimal des chemins joignant u à v et $\mathrm{C}(v)$ réalise un tel chemin (ou alors, $\mathrm{T}(v) = +\infty$ si $u \nrightarrow v$). L'idée est de construire de proche en proche un ensemble $X \subseteq V$ de manière telle qu'un chemin de poids minimal de u à $v \in X$ passe uniquement par des sommets de X. L'ensemble X est initialisé à $\{u\}$ et à chaque étape, on ajoute un sommet à l'ensemble.
(**Algorithme de Dijkstra**). Les données sont un digraphe simple $G = (V, E)$ pondéré par une fonction $p : V \times V \to \mathbb{R}^+ \cup \{+\infty\}$ et un sommet u. Dans cet algorithme, la notation $[\mathrm{C}(v), y]$

Algorithm 2 Algorithme de Dijkstra

```
1 : Entrée : un graphe pondéré G = (V, E), une source s
2 : Sortie : les distances minimales d(v) pour tout v ∈ V
3 : Initialiser d(v) ← ∞ pour tout v ∈ V, sauf d(s) ← 0
4 : Initialiser un ensemble Q ← V (sommets non encore traités)
5 : Tant que Q ≠ ∅ faire
6 :     Choisir u ∈ Q tel que d(u) soit minimal
7 :     Retirer u de Q
8 :     Pour chaque voisin v de u avec arête (u, v) de poids w(u, v) faire
9 :         Si d(v) > d(u) + w(u, v) alors
10 :            d(v) ← d(u) + w(u, v)
11 :        Fin Si
12 :    Fin Pour
13 : Fin Tant que
14 : Retourner d
```

représente la liste $\mathrm{C}(v)$ à laquelle on ajoute un élément y. Intuitivement, lorsqu'on ajoute un sommet v à X, on regarde s'il est avantageux pour les sommets y ne se trouvant pas dans X de passer par ce sommet v nouvellement ajouté à X. Si tel est le cas, on met à jour les informations concernant y.

8. En effet, passer par une boucle de poids $\alpha > 0$ ne ferait qu'augmenter inutilement le poids de ce chemin. De même, si plusieurs arcs joignent deux sommets, il suffit de conserver l'arc de poids minimal.

9. Comme nous l'avons définie, la distance n'a de sens que sur des digraphes fortement connexes.

Exemple 2.2.10 *Voici une application de l'algorithme de Dijkstra au graphe représenté à la figure 2.21. Pour l'initialisation, prenons* $\mathrm{X} = \{a\}$ *et on a*

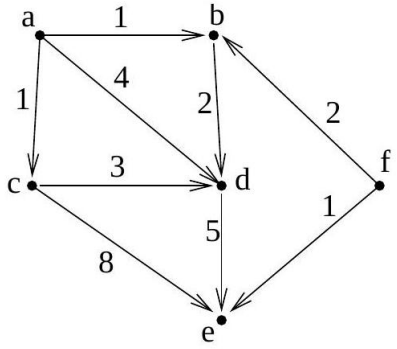


FIGURE 2.21 – Un digraphe simple pondéré.

v	a	b	c	d	e	f
$\mathrm{T}(v)$	*0*	*1*	*1*	*4*	$+\infty$	$+\infty$
$\mathrm{C}(v)$	(a,a)	(a,b)	(a,c)	(a,d)	(a,e)	(a,f)

Si le sommet b est choisi (on a le choix entre b et c), on a $\mathrm{X} = \{a,b\}$ et le tableau est mis à jour :

v	a	b	c	d	e	f
$\mathrm{T}(v)$	0	1	1	3	$+\infty$	$+\infty$
$\mathrm{C}(v)$	(a,a)	(a,b)	(a,c)	(a,b,d)	(a,e)	(a,f)

.

A l'étape suivante, on est forcé de choisir c. Ainsi, $\mathrm{X} = \{a,b,c\}$ et le tableau devient :

v	a	b	c	d	e	f
$\mathrm{T}(v)$	0	1	1	3	9	$+\infty$
$\mathrm{C}(v)$	(a,a)	(a,b)	(a,c)	(a,b,d)	(a,c,e)	(a,f)

.

A présent, d est sélectionné, $\mathrm{X} = \{a,b,c,d\}$ et la valeur de $\mathrm{C}(e)$ peut être améliorée,

v	a	b	c	d	e	f
$\mathrm{T}(v)$	0	1	1	3	8	$+\infty$
$\mathrm{C}(v)$	(a,a)	(a,b)	(a,c)	(a,b,d)	(a,b,d,e)	(a,f)

Ensuite, e et f sont choisis successivement mais le tableau ne change plus. Par exemple, on en conclut que le plus court chemin joignant a à e passe par b et d et son poids vaut 8.

Exemple 2.2.11 *Application à un graphe orienté en forme d'hexagone (voir figure 2.22.)*

Déroulement depuis A :

— Initialisation : A(0), B(∞), C(∞), D(∞), E(∞), F(∞)

9. [17] On suppose que $r < +\infty$, si $r \in \mathbb{R}$ et que $+\infty \leq +\infty$.

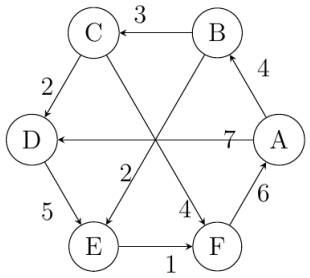


FIGURE 2.22 – Graphe en forme d'hexagone

— Traitement A : B(4), D(7), F(6)

— Traitement B : E(4+2=6)

— Traitement F : A déjà traité

— Traitement D : E(min(6,7+5=12)=6)

— Traitement E : F(min(6,6+1=7)=6)

— Traitement C : depuis B(3) ou F(6+4=10) C(7).

Résultats : A(0), B(4), C(7), D(7), E(6), F(6).

Démonstration. Il est clair que l'algorithme s'achève toujours puisqu'à chaque itération de la boucle, un nouvel état est ajouté à X. Il nous suffit donc de vérifier qu'il s'achève avec le résultat attendu. Pour ce faire, on procède par récurrence sur $\#X$ et on vérifie les deux assertions suivantes pour toutes les valeurs de $\#X$, d'où le résultat annoncé quand $X = V$.
i) Pour tout $v \in X$, $\mathrm{T}(v)$ est le poids minimal de tous les chemins joignant u à v.
ii) Pour tout $v \notin X$, $\mathrm{T}(v)$ est le poids minimal des chemins joignant u à v qui, à l'exception de v, passent uniquement par des sommets de X.
Pour $\#X = 1$, c'est immédiat car $X = \{u\}$ et l'initialisation effectuée dans l'algorithme correspond aux assertions. Supposons le résultat vérifié pour $\#X = n$ et vérifions-le pour $\#X = n + 1, 1 \leq n < \#V$. Lorsque X contient n sommets, l'algorithme stipule de lui ajouter un sommet v ayant une valeur $\mathrm{T}(v)$ minimale parmi les sommets n'appartenant pas à X. Par l'hypothèse de récurrence ii), puisqu'à ce stade $\#X = n$ et $v \notin X$, $\mathrm{T}(v)$ est le poids minimal des chemins joignant u à v qui, à l'exception de v, passent uniquement par des sommets de X. Nous ajoutons à présent v à X pour obtenir un ensemble de taille $n+1$. Procédons par l'absurde et supposons que $\mathrm{T}(v)$ n'est pas le poids minimal des chemins joignant u à v (c'est-à-dire que l'assertion i) n'est pas vérifiée). Par conséquent, il existe un sommet $y \notin X$ et un chemin de u à v passant par y dont le poids est strictement inférieur à $\mathrm{T}(v)$. De là, on en conclut que $\mathrm{T}(y) < \mathrm{T}(v)$, ce qui est impossible au vu du choix du sommet v au sein de l'algorithme. Pour la seconde assertion, on procède de manière analogue en utilisant cette fois la dernière ligne de l'algorithme pour obtenir une contradiction. Plus précisément, passons d'un ensemble X à n sommets à un ensemble X' à $n + 1$ sommets en lui ajoutant un sommet v. Supposons à présent que ii) n'est plus satisfait et qu'il existe donc un sommet $y \notin X'$ tel que $\mathrm{T}(y)$ est strictement supérieur au poids minimal des chemins joignant u à y qui, à l'exception de y, passent uniquement par des sommets de X'. Or, par hypothèse de récurrence, nous savons que l'assertion ii) était satisfaite pour $\#X = n$. Autrement dit, avant d'ajouter le sommet v, $\mathrm{T}(y)$ était minimal pour les chemins joignant u à y qui, à l'exception de y, passent uniquement par des sommets de X. Ainsi, en ajoutant le sommet v à X, on aurait remplacé $\mathrm{T}(y)$ par une valeur supérieure, ce qui est

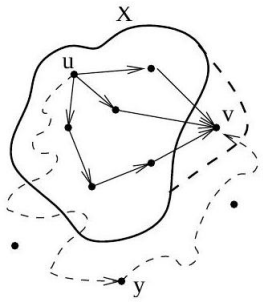


FIGURE 2.23 – Illustration de l'algorithme de Dijkstra.

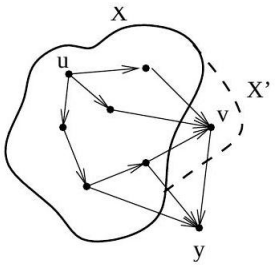


FIGURE 2.24 – llustration de l'algorithme de Dijkstra.

en contradiction avec les prescriptions de l'algorithme (à la dernière ligne, on spécifie de remplacer T(y) uniquement lorsqu'il est préférable de passer par v). □

Remarque 2.2.12 *On peut facilement voir que l'algorithme de Dijkstra a une complexité temporelle en* $\mathcal{O}((\#V)^2)$. *Avec une implémentation minutieuse, utilisant les listes d'adjacence et les files de priorité, on obtient même une complexité en* $\mathcal{O}((\#E + \#V)\log \#V)$.

Remarque 2.2.13 *Le routage des données entre les réseaux d'un internet est l'une des applications où les plus courts chemins jouent un rôle important. Le routage est le processus consistant à prendre des décisions sur la façon de déplacer des données d'un point à un autre. Dans un internet, il est effectué en propageant de petites portions de données, les paquets, le long de points interconnectés, les passerelles. Lorsque chaque paquet passe par une passerelle, un routeur étudie où le paquet doit se rendre et décide alors de la passerelle suivante où l'envoyer. Le but de chaque routeur est de propager un paquet vers un point de plus en plus près de sa destination finale.*

Afin de rapprocher un paquet de sa destination, chaque routeur gère des informations sur la structure, ou topologie, de l'internet. Ces informations sont stockées dans une table de routage, contenant une entrée pour chaque passerelle que le routeur sait atteindre. Chaque entrée indique la passerelle suivante à laquelle envoyer les paquets destinés à une autre passerelle que le routeur lui-même.
Pour que les paquets soient continuellement envoyés par la meilleure route possible, les routeurs mettent régulièrement à jour leurs tables de routage afin de tenir compte des changements dans l'internet. Dans l'un des types de routage, appelé routage SPF (Shortest Path First), chaque routeur gère sa propre carte de l'internet afin de mettre à jour sa table de routage en calculant les plus courts chemins entre lui et les autres destinations. Sa carte est un graphe orienté et pondéré, dont les sommets sont les passerelles et les arcs les connexions entre celles-ci. Chaque arc est pondéré par les dernières performances constatées sur la connexion.

2.2.2 Graphes et chemins eulériens

La définition suivante est valable aussi bien dans le cas de graphes orientés que non orientés. Nous supposerons à chaque fois qu'il sera question de problèmes eulériens être en présence de graphes connexes.

Définition 2.2.14 *Un chemin (resp. un circuit) d'un multi-graphe G est eulérien s'il passe une et une seule fois par chaque arête/arc de G. (Un tel chemin (resp. circuit) peut bien évidemment passer plus d'une fois par un même sommet.) Autrement dit, un chemin (resp. un circuit) eulérien est une piste (resp. une piste fermée) passant par chaque arête/arc de G. Un multi-graphe eulérien est un graphe qui possède un circuit eulérien.*

Dans le problème consistant à déterminer si un graphe G possède ou non un chemin eulérien, l'existence de boucles au sein de G n'a aucune importance. En effet, soient G un graphe et G' le graphe obtenu en supprimant les boucles de G. Il est évident que G possède un chemin ou un circuit eulérien si et seulement si G' en possède un.
Considérons le cas des multi-graphes finis non orientés. Il est évident que pour que G possède un circuit eulérien, i.e., pour que G soit eulérien, il est nécessaire que G soit connexe et que le degré de chaque sommet soit pair. Comme le montre le résultat suivant, ces conditions sont également suffisantes. On peut donc constater que le fait d'être eulérien est une propriété "locale" (elle ne fait intervenir que le degré de chaque sommet pris "isolément").

Théorème 2.2.15 *Un multi-graphe fini non orienté connexe $G = (V, E)$ possède un circuit eulérien si et seulement si le degré de chaque sommet est pair.*

Démonstration. Supposons donc que chaque sommet de G est de degré pair. Débutons la construction d'une piste avec un sommet a_1 de G. A chaque étape $i \geq 1$ de cette construction, on choisit un sommet a_{i+1} de manière telle qu'une arête $\{a_i, a_{i+1}\} \in E$ est sélectionnée parmi les $\#E - i + 1$ arêtes non déjà sélectionnées. Puisque chaque sommet est de degré pair, cette sélection est toujours possible ("lorsqu'on aboutit dans un sommet, on peut toujours en repartir"). Puisque le graphe est fini, cette procédure s'achève toujours. On dispose alors d'une piste P joignant a_1 à un certain sommet a_ℓ. En fait, on peut supposer que cette piste est fermée, i.e., $a_\ell = a_1$. En effet, si a_ℓ diffère de a_1, puisque le degré de chaque sommet est pair, on peut étendre la piste en ajoutant une arête $\{a_\ell, a_{\ell+1}\}$. En continuant de la sorte, on épuise les sommets jusqu'à revenir en a_1. Si la piste fermée P est un circuit eulérien, le théorème est démontré. Sinon, il existe un sommet b de P qui est l'extrémité d'un nombre pair d'arêtes n'apparaissant pas dans P. (Une illustration est donnée à la figure 2.25. Depuis b, il est possible de construire une piste fermée Q formée uniquement d'arêtes n'apparaissant pas dans P. (On utilise la même procédure que précédemment ; il est clair que le degré de chaque sommet est encore pair.) De cette façon, nous avons étendu la piste P en une piste plus longue $P \cup Q$ (couvrant un plus grand nombre d'arêtes). On obtient alors un circuit eulérien en répétant cette étape un nombre fini de fois. □

Corollaire 2.2.16 *Le problème des sept ponts de Königsberg donné dans l'exemple 2.1.22 n'admet pas de solution.*

Démonstration. C'est immédiat, le graphe de la figure 2.12 possède un sommet de degré 5 et 3 sommets de degré 3. □

Corollaire 2.2.17 *Un multi-graphe non orienté connexe possède un chemin eulérien joignant deux sommets a et b si et seulement si a et b sont les deux seuls sommets de degré impair.*

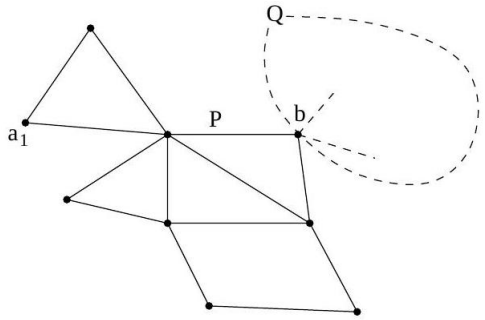


FIGURE 2.25 – Construction d'un circuit eulérien.

Démonstration. Pour se ramener au théorème précédent, il suffit de considérer le graphe G auquel on ajoute une arête supplémentaire joignant les sommets a et b. □

Exemple 2.2.18 *Fort des résultats précédents, on peut par exemple répondre à la question : "le dessin suivant peut-il ou non être tracé d'un seul trait, sans lever le crayon et sans repasser deux fois par un trait déjà tracé ?". C'est une simple application du corollaire précédent. Nous présentons*

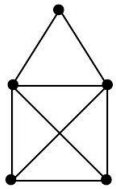


FIGURE 2.26 – Une maison à tracer d'un seul trait.

succintement l'algorithme de Fleury permettant de construire un circuit eulérien (à supposer qu'un tel circuit existe).

Algorithm 3 Algorithme de Fleury

1 : Entrée : un graphe connexe $G = (V, E)$ contenant un cycle eulérien ou semi-eulérien
2 : Sortie : un chemin ou cycle eulérien
3 : Choisir un sommet de départ v (sommet de degré impair si semi-eulérien, sinon n'importe lequel)
4 : **Tant que** il reste des arêtes dans G **faire**
5 : Choisir une arête $e = (v, u)$ incidente à v qui n'est pas un pont, sauf si aucune autre option nexiste
6 : Ajouter e au chemin
7 : Supprimer e du graphe
8 : Poser $v \leftarrow u$
9 : **Fin Tant que**

Lalgorithme de Fleury est efficace pour les petits graphes. La détection des ponts peut être coûteuse (nécessite une vérification de connexité à chaque étape).
Pour les graphes larges, lalgorithme 4 de Hierholzer est préféré (plus efficace).

9. Il est clair que pour trouver la solution à ce problème, il faut débuter son dessin par un des deux coins inférieurs. Ce sont les deux seuls sommets de degré impair.

Algorithm 4 Algorithme de Hierholzer

```
Entrée : graphe G = (V, E) eulérien
Sortie : cycle eulérien
Choisir un sommet de départ v
Former un cycle C en suivant des arêtes non encore utilisées jusqu'à revenir à v
Tant que il reste des arêtes non utilisées faire
    Choisir un sommet u de C qui a encore des arêtes non utilisées
    Former un cycle C' à partir de u
    Fusionner C' dans C
Fin Tant que
Retourner C
```

Théorème 2.2.19 *Un multi-graphe fini orienté simplement connexe $G = (V, E)$ possède un circuit eulérien si et seulement si le demi-degré entrant de chaque sommet est égal à son demi-degré sortant.*

Démonstration. Laissée à titre d'exercices. On adaptera facilement le raisonnement de la preuve du théorème 2.2.15.

Corollaire 2.2.20 *Un multi-graphe fini orienté connexe $G = (V, E)$ possède un chemin eulérien si et seulement si il existe deux sommets v_0 et v_1 tel que*

- *pour tout* $v \in V \setminus \{v_0, v_1\}, d^-(v) = d^+(v)$,
- $d^+(v_0) = d^-(v_0) + 1$,
- $d^-(v_1) = d^+(v_1) + 1$.

Démonstration. Laissée à titre d'exercices. □

2.2.3 Connexité des graphes non orientés.

Nous venons de voir qu'une condition nécessaire pour qu'un graphe soit eulérien est qu'il soit connexe. Nous allons donc fournir un algorithme naïf permettant de décider si un graphe non orienté est connexe. Il est clair que lorsqu'on s'intéresse à cette question, il suffit de considérer des graphes simples. (En effet, l'existence de boucles ou d'arêtes multiples n'a pas d'incidence sur la connexité du graphe.)

La donnée fournie à cet algorithme est un graphe non orienté $G = (V, E)$, l'ensemble des sorties possibles est {oui, non}.

Il s'agit d'un algorithme "tache d'huile" ou "glouton" (on détermine de proche en proche, les sommets accessibles depuis le sommet v_0). La variable Composante est destinée à contenir la composante connexe du sommet v_0. La variable New contient les sommets qui, à l'étape en cours d'exécution, sont nouvellement déterminés comme appartenant à la composante connexe. Il s'agit donc de deux variables de type ensembliste. Lorsqu'il n'y a plus de nouveaux sommets à ajouter à la liste des sommets appartenant à la composante connexe, l'algorithme s'achève. Il reste alors à déterminer si cette composante connexe contient ou non tous les sommets de G. En particulier, à la fin de cet algorithme, la variable Composante est égale à la composante connexe de v_0. L'emploi de cet algorithme est facilité par la construction préalable d'un dictionnaire des voisins. Il s'agit d'une liste associant à chaque sommet v, l'ensemble $\nu(v)$ de ses voisins.

Algorithm 5 Breadth-First Search (BFS)

Choisir au hasard un sommet $v_0 \in V$
Composante := $\{v_0\}$, New:= $\{v_0\}$
Tant que New $\neq \emptyset$ **faire**
  Voisins:= $\emptyset$
  Pour tout sommet v appartenant à New **faire**
    Voisins:= Voisins $\cup \nu(v)$
  Fin Pour
  New:= Voisins\Composante
  Composante:= Composante$\cup$New
Fin Tant que
Si Composante=V **alors**
  alors sortie : "oui, G connexe"
  sinon sortie : "non, G non connexe"
Fin Si

Exemple 2.2.21 *Considérons le graphe simple représenté à la figure 2.27 et appliquons-lui l'algorithme 5. Les résultats de l'algorithme se trouvent dans le tableau 2.2. Le tableau 2.1 donne le dictionnaire des voisins*

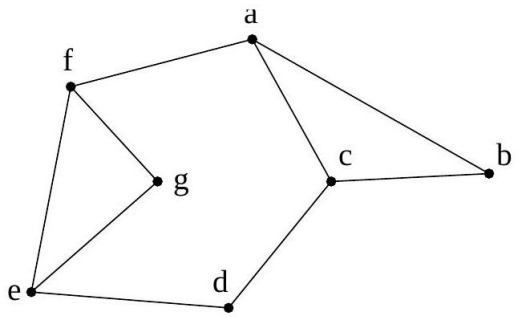


FIGURE 2.27 – Un graphe simple (connexe).

construit une fois pour toutes.

v	$\nu(v)$
a	$\{b, c, f\}$
b	$\{a, c\}$
c	$\{a, b, d\}$
d	$\{c, e\}$
e	$\{d, f, g\}$
f	$\{a, e, g\}$
g	$\{e, f\}$

TABLE 2.1 – Dictionnaire des voisins.

Composante	New	Voisins
$\{c\}$	$\{c\}$	$\emptyset$ $\nu(c) = \{a, b, d\}$
$\{c\} \cup \{a, b, d\}$ $= \{a, b, c, d\}$	$\{a, b, d\} \setminus \{c\}$ $= \{a, b, d\}$	$\emptyset$ $\nu(a) \cup \nu(b) \cup \nu(d) = \{a, b, c, e, f\}$
$\{a, b, c, d\} \cup \{e, f\}$ $= \{a, b, c, d, e, f\}$	$\{a, b, c, e, f\} \setminus \{a, b, c, d\}$ $= \{e, f\}$	$\emptyset$ $\nu(e) \cup \nu(f) = \{a, d, e, f, g\}$
$\{a, b, c, d, e, f\} \cup \{g\}$ $= \{a, b, c, d, e, f, g\}$	$\{a, d, e, f, g\} \setminus \{a, b, c, d, e, f\}$ $= \{g\}$	$\emptyset$ $\nu(g) = \{e, f\}$
$\{a, b, c, d, e, f, g\} \cup \{e, f\}$ $= \{a, b, c, d, e, f, g\}$	$\{e, f\} \setminus \{a, b, c, d, e, f, g\}$ $= \emptyset$	

TABLE 2.2 – Application de l'algorithme de parcours en largeur (BFS) pour tester la connexité

2.2.4 Décomposition en composantes fortement connexes

La détermination de la composante fortement connexe contenant un sommet donné v_0 s'inspire de l'algorithme 5. Il est clair que, pour ce problème, il suffit de considérer le cas de graphes orientés simples. Déterminer si un graphe est connexe, autrement dit s'il possède une seule composante connexe, est l'une des deux étapes permettant de décider si un graphe donné possède ou non une structure d'arbre (comme nous le verrons, les arbres sont particulièrement importants à bien des égards).

Exemple 2.2.22 *La clôture réflexive et transitive de l'application succ est définie comme suit,*

$$\mathrm{succ}^*(v) := \bigcup_{j=0}^{\infty} \mathrm{succ}^j(v) \qquad où \begin{cases} \mathrm{succ}^0(v) = v \\ \mathrm{succ}^{j+1}(v) = \mathrm{succ}\left(\mathrm{succ}^j(v)\right). \end{cases}$$

Si le graphe G est fini, il est clair que

$$\mathrm{succ}^*(v) = \bigcup_{j=0}^{\#E} \mathrm{succ}^j(v)$$

De même, s'il existe $k < \#E$ tel que $\mathrm{succ}^k(v) = \mathrm{succ}^{k+1}(v)$, alors

$$\mathrm{succ}^*(v) = \bigcup_{j=0}^{k} \mathrm{succ}^j(v).$$

Avec les notations précédentes, pour tous $a, b \in V$,

$$a \to b \Leftrightarrow b \in \mathrm{succ}^*(a).$$

Remarque 2.2.23 *On définit de manière semblable la clôture réflexive et transitive de l'application pred. On a bien évidemment, pour tous sommets a et b,*

$$b \to a \Leftrightarrow b \in \text{pred}^*(a).$$

Ainsi,

$$a \leftrightarrow b \Leftrightarrow b \in \text{succ}^*(a) \cap \text{pred}^*(a).$$

L'algorithme 5 peut facilement être adapté pour calculer succ(a) (resp. pred(a)). Il suffit principalement d'initialiser les variables Composante et New à $\{a\}$ et de remplacer $\nu(v)$ par succ (v)(resp. pred (v)). En recherchant ainsi l'intersection des deux ensembles, on détermine la composante f. connexe du sommet a. Si cette composante est égale à l'ensemble des sommets, alors le graphe est f . connexe.

Définition 2.2.24 *Soit $G = (V, E)$ un graphe orienté. On construit un nouveau graphe orienté, appelé graphe acyclique des composantes* [21] *ou encore condensé de G, dont les sommets sont les composantes f . connexes de G. Un arc joint deux composantes f . connexes A et B, s'il existe $a \in A$ et $b \in B$ tels que $a \to b$.*

Remarque 2.2.25 *Bien évidemment, s'il existe $a \in A$ et $b \in B$ tels que $a \to b$, alors il n'existe aucun $a' \in A$ ni aucun $b' \in B$ tels que $b' \to a'$. En effet, si tel était le cas, $A \cup B$ serait alors une composante f. connexe de G. Ceci est impossible vu la maximalité des composantes connexes. D'une manière générale, le graphe des composantes ne contient pas de cycle.*

Exemple 2.2.26 *La figure 2.28 représente un digraphe et ses composantes f. connexes ainsi que le graphe des composantes correspondant.*

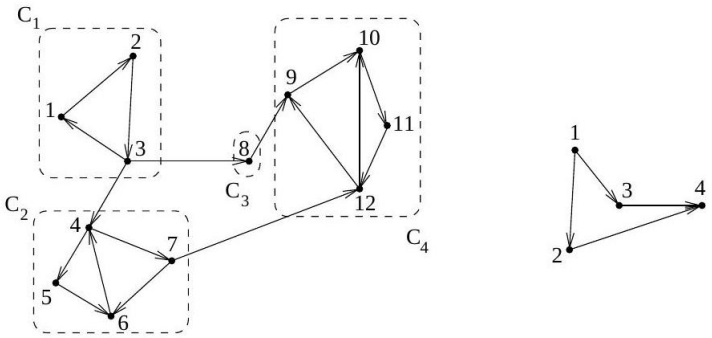


FIGURE 2.28 – Digraphe et graphe des composantes.

Le résultat suivant donne un lien entre la simple connexité et la forte connexité.

Lemme 2.2.27 *Soit G un digraphe (simple) tel que pour tout $v \in V$, $d^+(v) = d^-(v)$. Alors, G est f. connexe si et seulement si il est s. connexe.*

Démonstration. Il est clair que la f. connexité implique la s. connexité. Supposons que G est s. connexe, que pour tout $v, d^+(v) = d^-(v)$, mais que G n'est pas f. connexe. Notre but est d'obtenir une contradiction. Soient $C_1, \ldots, C_r$ les composantes f. connexes de G (où $r \geq 2$). Comme nous l'avons déjà remarqué plus haut (remarque 2.2.25), le condensé de G (ayant les C_i pour sommets) ne

possède pas de cycle. On en conclut [22] qu'il existe (au moins) une composante C_j telle que tout arc issu d'un sommet de C_j possède sa destination également dans C_j (dans l'exemple correspondant à la figure I.38, c'est le cas de la composante C_4). Par hypothèse, G est s. connexe. Il existe donc une autre composante C_i et un arc joignant un sommet x de C_i à un sommet y de C_j.

Dans le graphe $G' = (V', E')$ restreint à la composante C_j, i.e., dans le graphe où l'on considère uniquement les sommets de C_j et les arcs ayant leurs deux extrémités dans C_j, la "handshaking formula" est satisfaite, i.e.,

$$\sum_{v \in V'} \mathrm{d}^-(v) = \sum_{v \in V'} \mathrm{d}^+(v).$$

Il est sous-entendu que si le symbole de sommation fait référence à V', alors on considère le graphe G' correspondant. Revenons au graphe G de départ. Dans celui-ci, tout arc ayant son origine dans C_j a aussi son extrémité dans C_j. Ainsi, considérer le graphe G dans son entièreté ou sa restriction G' à C_j ne modifie pas le demi-degré sortant des sommets de C_j et on a

$$\sum_{v \in C_j} \mathrm{d}^+(v) = \sum_{v \in V'} \mathrm{d}^+(v).$$

Par contre, y est un sommet de C_j et nous savons qu'au moins un arc n'ayant pas son origine dans C_j aboutit en y. Autrement dit, nous avons cette fois,

$$\sum_{v \in C_j} \mathrm{d}^-(v) > \sum_{v \in V'} \mathrm{d}^-(v).$$

De là, on en conclut que

$$\sum_{v \in C_j} \mathrm{d}^-(v) > \sum_{v \in C_j} \mathrm{d}^+(v)$$

ce qui contredit l'hypothèse selon laquelle $d^+(v) = d^-(v)$ pour tout $v \in V$ et donc que $\sum_{v \in C_j} \mathrm{d}^-(v) = \sum_{v \in C_j} \mathrm{d}^+(v)$. □

Une preuve alternative bien plus courte mais exploitant un résultat précédent. En fait, la preuve ci-dessus reprend les mêmes raisonnements que ceux développés dans la preuve du théorème 2.2.15.

Démonstration. Supposons le graphe s. connexe. Puisqu'il est connexe et que pour pour tout $v \in V, d^+(v) = d^-(v)$, on peut appliquer le théorème 2.2.19. On conclut directement que G possède un circuit eulérien. Il est donc f. connexe. □

2.3 Sous-graphes

Définition 2.3.1 *Soient $G = (V, E)$ et $G' = (V', E')$ deux graphes (orientés ou non). Le graphe G' est un sous-graphe de G si*

- $V' \subseteq V$,
- $E' \subseteq E \cap (V' \times V')$.

Ainsi, G' est un sous-graphe de G s'il est obtenu en enlevant à G certains sommets et/ou certains arcs ou arêtes. En particulier, si on enlève un sommet v de G, il faut nécessairement enlever tous les arcs (ou arêtes) incidents à v. Par contre, pour construire G', on peut très bien enlever un arc (ou une arête) de G sans pour autant enlever le moindre sommet de G. Le graphe G' est un sous-graphe

propre de G si $E' \subsetneq E$ ou $V' \subsetneq V$. Dans le premier sous-graphe de la figure 2.29, on a enlevé uniquement certaines arêtes. Dans le second, on a enlevé un sommet et les arêtes adjacentes.

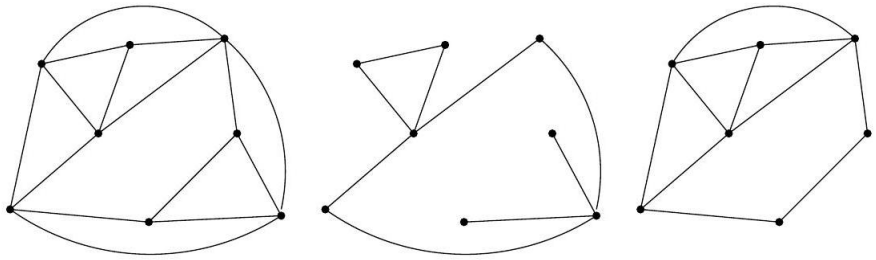


FIGURE 2.29 – Un graphe et deux sous-graphes.

Soient $v \in V$ et $e \in E$. On note $G - e$ (resp. $G - v$) le sous-graphe G' de G obtenu en supprimant l'arc (ou l'arête) e (resp. le sous-graphe G' obtenu en supprimant le sommet v et les arcs (ou les arêtes) adjacents). Par analogie, on notera $G = G' + e$ (resp. $G = G' + v$), le graphe obtenu par adjonction à G' d'une arête ou d'un sommet.
On peut bien évidemment étendre ces notations à un ensemble fini de sommets. Ainsi, si $W = \{v_1, \ldots, v_k\} \subseteq V$, alors $G - W$ est le sous-graphe

$$(\cdots((G - v_1) - v_2) \cdots - v_{k-1}) - v_k := G - v_1 - \cdots - v_k.$$

On procède de même pour un ensemble fini d'arcs (ou d'arêtes) et on introduit une notation $G - F$ pour un sous-ensemble F de E.
Soit $W \subseteq V$. Le sous-graphe de G induit par W est le sous-graphe $G' = (W, E')$ avec $E' = E \cap (W \times W)$.
Si $W \subseteq V$ est tel que le sous-graphe induit par W ne contient aucune arête, alors les sommets de W sont dits indépendants. Le nombre maximal de sommets indépendants de G est noté $\alpha(G)$.

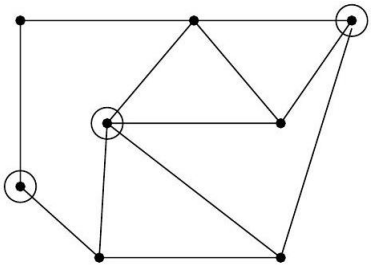


FIGURE 2.30 – Des sommets indépendants.

Définition 2.3.2 *Soient $G = (V, E)$ un graphe et $G' = (V', E')$ un de ses sous-graphes. On dit que G' est un sous-graphe couvrant G, si $V' = V$ et si*

$$\forall v \in V, \exists z \in V : \{z, v\} \in E'.$$

On dira que E' est une couverture (des sommets) de G. Autrement dit, tout sommet de G est une extrémité d'une arête de E'. Comme nous le verrons bientôt, on s'intéressera en particulier aux sousgraphes couvrants qui sont des arbres. Dans ce cas, on parlera naturellement de sous-arbre couvrant.

Exemple 2.3.3 *Le premier sous-graphe de la figure 2.29 est couvrant. Un exemple de sous-arbre couvrant a été donné à la figure 2.31.*

Exemple 2.3.4 *(Arbre couvrant). Une société de téléphonie souhaite câbler entièrement, au moyen de nouvelles fibres optiques, une ville en minimisant le nombre de connexions à réaliser. Le nouveau cablage s'appuie sur le réseau électrique déjà existant et bien évidemment, tous les points de la ville doivent être désservis. La figure 2.31 représente le réseau actuel de la ville et ses connexions. A droite, se trouve les sélections envisagées. La question générale qui est posée est de rechercher un sous-graphe*

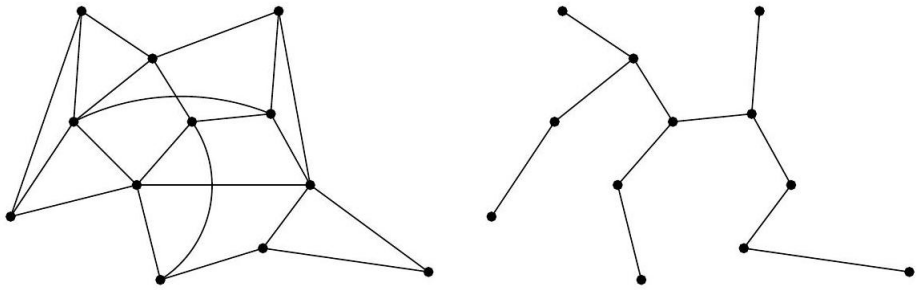


FIGURE 2.31 – Sous-graphe couvrant.

(ou un sous-arbre) couvrant dans un graphe donné. On peut aussi envisager une version pondérée dans laquelle chaque arc aurait un coût et on rechercherait un sous-graphe (ou un sous-arbre) couvrant de poids minimal.

2.4 Arbres

Les graphes connexes acycliques jouent un rôle prépondérant dans diverses applications (cf. l'exemple 2.3.4 ou encore de nombreux exemples tirés de l'informatique : arbres binaires de recherche, bases de données, etc...).

Définition 2.4.1 *Un graphe simple non orienté $A = (V, E)$ est un arbre s'il est connexe et sans cycle (sous-entendu, un "véritable" cycle : une piste fermée, pas un cycle "artificiel" comme un trivial $(\{a, b\}, \{b, a\})$ qui est un cycle de longueur 2 ; le fait d'imposer l'absence d'une piste fermée évite de telles arêtes doublées et on pourrait de manière équivalente imposer l'absence de circuit simple). Une forêt est un graphe simple non orienté dont chaque composante connexe est un arbre. Un arbre $A = (V, E)$ est qualifié de n-aire si pour tout sommet $v \in V, \deg(v) \leq n$.*

Proposition 2.4.2 *Soit $G = (V, E)$ un arbre ayant n sommets.*

- *Toute paire de sommets distincts de G est connectée par exactement un chemin simple.*
- *Soit $e \in (V \times V) \backslash E$ qui n'est pas une boucle. Le graphe $G + e$ contient un cycle (i.e., une piste fermée), c'est-à-dire, $G + e$ n'est plus un arbre.*
- *Le graphe G a exactement $n - 1$ arêtes.*

Démonstration. C'est immédiat. □

On pourra en particulier observer que tout arbre est 1 -connexe, la réciproque étant bien évidemment fausse.

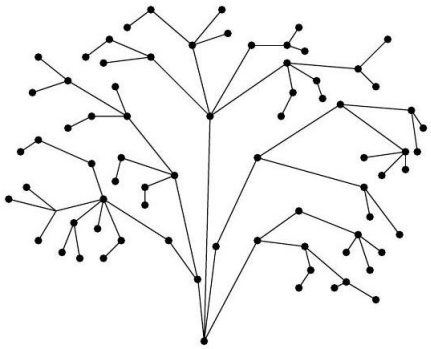


FIGURE 2.32 – Un arbre.

Proposition 2.4.3 *Un graphe $G = (V, E)$ simple connexe est un arbre si et seulement si chacune de ses arêtes est une arête de coupure.*

Démonstration. Admise. □

Proposition 2.4.4 *Tout graphe connexe possède un sous-arbre couvrant.*

Démonstration. Soient $G = (V, E)$ un graphe connexe et $C = (V, E')$ un sous-graphe couvrant connexe minimal (i.e., on ne peut pas remplacer E' par un sous-ensemble strict tout en conservant la connexité de C). Vu la minimalité de C, chacune de ses arêtes est une arête de coupure de C. Par la proposition précédente, on en conclut que C est un arbre. □

Corollaire 2.4.5 *Si $G = (V, E)$ est un graphe (simple non orienté) connexe, alors $\#E \geq \#V - 1$.*

Démonstration. Par le corollaire précédent, G possède un sous-arbre couvrant $C = (V, E')$. De là, il vient

$$\#E \geq \#E' = \#V - 1$$

où pour la dernière égalité, on a utilisé la proposition 2.4.2. □

Définition 2.4.6 *Un arbre $A = (V, E)$ dans lequel on a privilégié un sommet v_0 est appelé arbre pointé. On le notera (A, v_0). Le sommet v_0 est parfois appelé la racine de l'arbre.*

Pour un arbre pointé (A, v_0), les sommets de A peuvent être ordonnés suivant leur distance à v_0. Si v est un sommet tel que $d(v_0, v) = i$, on dira que v est un sommet de niveau i. Un arbre pointé a été représenté à la figure 2.33.

Si v est un sommet de niveau i et si tous ses voisins sont de niveau $i - 1$, on dit alors que v est une feuille de l'arbre. A la figure 2.33, les feuilles ont été marquées d'un cercle. La hauteur d'un arbre est le niveau maximal de ses feuilles. L'arbre de la figure 2.34 est un arbre de hauteur 4.

Remarque 2.4.7 *Pointer un arbre définit naturellement une orientation des arêtes de l'arbre. En effet, on peut orienter les arcs de façon à ce qu'ils joignent les sommets de niveau i aux sommets de niveau $i + 1$. Dans ce cadre, on parle souvent des fils (resp. du père) d'un nœud v pour désigner ses successeurs (resp. son unique prédécesseur). Les descendants (resp. ancêtres) de v désignent les éléments de* $\text{succ}^*(v)$ $(\text{resp. } \text{pred}^*(v))$.

9. En anglais, *rooted tree.*

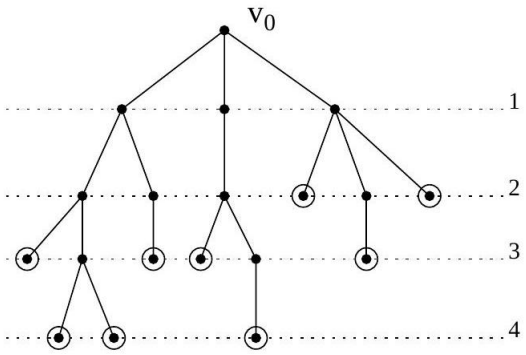


FIGURE 2.33 – Un arbre pointé.

Définition 2.4.8 *Un arbre pointé est k-aire si tout sommet a au plus k fils. Si $k = 2$, on parle naturellement d'arbre binaire. Un arbre k-aire de hauteur n possède au plus*

$$1 + k + \cdots + k^n = \frac{k^n - 1}{k - 1}$$

sommets. S'il en possède exactement ce nombre, on parle d'arbre k-aire complet.

2.4.1 Parcours d'arbres

Un parcours d'un arbre est une façon d'en ordonner les nœuds. On supposera implicitement que les fils d'un nœud v_i sont ordonnés $v_{i,1}, \ldots, v_{i,k_i}$ et que cet ordre est connu et fixé une fois pour toutes. On distingue trois types de parcours en profondeur : les parcours préfixe, infixe et suffixe. Soit (A, v_0) un arbre pointé. Pour le parcours préfixe, on parcourt d'abord la racine puis on parcourt, de manière récursive et dans l'ordre, les sous-arbres pointés de racine respective $v_{0,1}, \ldots, v_{0,k_0}$. Pour le parcours suffixe, on parcourt d'abord, de manière récursive et dans l'ordre,
les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0. Pour le parcours infixe, nous supposerons disposer d'un arbre binaire. (On peut donc parler du sous-arbre de gauche et du sous-arbre de droite.) On parcourt d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droite

Exemple 2.4.9 *Considérons l'arbre binaire pointé donné à la figure 2.34. Les fils d'un nœud sont ordonnés par l'ordre alphabétique de leur label.*

Si on parcourt cet arbre de manière préfixielle, on obtient la suite :

$$a, b, d, g, h, l, c, e, i, f, j, k.$$

Pour un parcours suffixe, on a

$$g, l, h, d, b, i, e, j, k, f, c, a.$$

Enfin, pour un parcours infixe, on obtient

$$g, d, l, h, b, a, i, e, c, j, f, k.$$

On rencontre parfois des parcours en largeur. Dans ce cas, on parcourt les nœuds de l'arbre pointé par niveau croissant. Sur l'exemple précédent, on a simplement l'ordre $a, b, c, \ldots, k, l$.

9. Si dans un arbre binaire, un sommet n'a qu'un seul fils, on prend par convention que le sous-arbre de droite de ce sommet est vide.

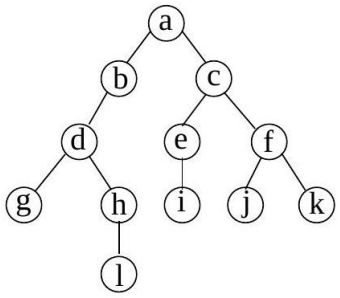


FIGURE 2.34 – Un arbre à parcourir.

Remarque 2.4.10 *On peut également définir un parcours en profondeur d'un graphe quelconque. Bien qu'il ne s'agisse pas d'un problème concernant spécifiquement les arbres, nous pensons qu'il s'agit du moment opportun pour le définir. Soit $G = (V, E)$ un graphe (orienté ou non) que l'on peut supposer simple et connexe. Un parcours en profondeur de G est défini récursivement comme suit. Sélectionner un sommet v_0. A l'étape $k \geq 1$, on choisit un voisin de v_{k-1} qui n'a pas encore été sélectionné. Si un tel voisin n'existe pas, on cherche dans l'ordre, un voisin non sélectionné de $v_{k-2}, \ldots, v_0$.*

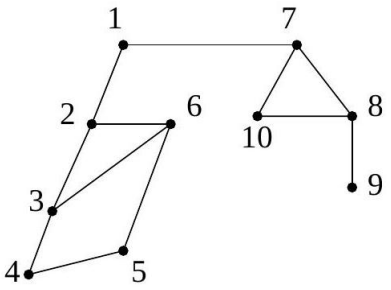


FIGURE 2.35 – Exemple de parcours en profondeur d'un graphe.

2.4.2 Recherche d'arbre couvrant minimum : Algorithme de Kruskal

2.4.3 Exercices

2.5 Un peu de théorie algébrique des graphes

Dans cette section, nous ne faisons qu'esquisser quelques résultats mettant en évidence les liens entre la théorie des graphes et l'algèbre linéaire (par exemple, on étudie le spectre des graphes réguliers ou bipartis). On y présente de manière pas très détaillée la théorie de Perron-Frobenius (sans démonstration du théorème principal) avec comme application directe, l'estimation du nombre de chemins de longueur n que l'on peut trouver dans un graphe à composantes connexes primitives. On présente également diverses questions relatives aux sous-arbres couvrants : nombre de tels sous-arbres, recherche d'un arbre de poids minimal, algorithme de Prim et de Kruskal. En particulier, cette section met en lumière quelques raisonnements de combinatoire énumérative.

2.5.1 Matrice d'adjacence

Définition 2.5.1 *Soit $G = (V, E)$ un multi-graphe non orienté dont les sommets sont ordonnés par $V = \{v_1, \ldots, v_n\}$. La matrice d'adjacence de G est la matrice $A(G)$ dont l'élément $[A(G)]_{i,j}$ est égal au nombre d'arêtes $\{v_i, v_j\}$ présentes dans $E, 1 \leq i, j \leq n$. (Pour rappel, E est en général un multi-ensemble.)*

Il s'agit donc d'une matrice symétrique à coefficients entiers naturels. Le polynôme caractéristique de G, noté $\chi_G(\lambda)$, est le polynôme caractéristique de sa matrice d'adjacence $A(G)$. Par abus de langage, on parlera des valeurs propres de G, étant sous-entendu qu'il s'agit des valeurs propres de $A(G)$. On parlera donc aussi du spectre de G. On peut remarquer que les éléments de la matrice d'adjacence d'un graphe simple appartiennent à $\{0, 1\}$ et que la trace de cette matrice vaut 0 .

Remarque 2.5.2 *En se rappelant quelques résultats du cours d'Algèbre II en deuxième année, on remarque que la matrice d'adjacence d'un graphe non orienté est toujours diagonalisable par une matrice orthogonale (pour chaque valeur propre, les multiplicités algébrique et géométrique coïncident) et que ses valeurs propres sont réelles.*

Exemple 2.5.3 *Considérons le graphe (simple) G de la figure 2.36. Avec les notations précédentes, on a aussi*

$$\chi_G(\lambda) = -\lambda^5 + 8\lambda^3 + 10\lambda^2 + \lambda - 2.$$

Proposition 2.5.4 *Deux graphes G_1 et G_2 sont isomorphes si et seulement si ils ont, à une permutation près, la même matrice d'adjacence.*

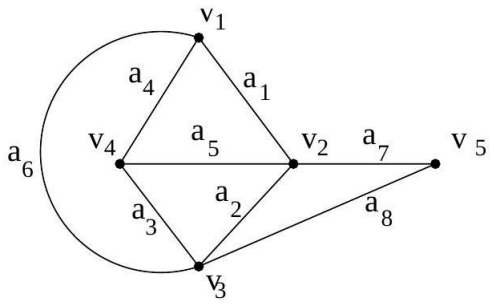


FIGURE 2.36 – Un graphe G et sa matrice d'adjacence.

$$A(G) = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Autrement dit, il existe une matrice de permutation P telle que

$$A(G_1) = P^{-1} A(G_2) P$$

Démonstration. C'est immédiat et ce résultat est même transposable au cas de graphes orientés. □

9. Une matrice de permutation est une matrice pour laquelle chaque ligne et chaque colonne contiennent exactement un 1, les autres entrées étant nulles.

Définition 2.5.5 *On peut aussi définir la matrice d'incidence "sommets/arêtes". Si* $V = \{v_1, \ldots, v_n\}$ *et* $E = \{e_1, \ldots, e_m\}$*, il s'agit d'une matrice* B *de dimension* $n \times m$ *telle que* $B_{i,j} = 1$ *si et seulement si* e_j *est incident à* v_i*.*

En poursuivant l'exemple précédent, cette matrice vaut ici

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Définition 2.5.6 *Dans un graphe simple, on appelle triangle, tout triplet d'arêtes distinctes deux à deux de la forme* $\{a, b\}, \{b, c\}, \{c, a\}$ *(i.e., tout circuit de longueur trois formé d'arêtes distinctes).*

Proposition 2.5.7 *Si le polynôme caractéristique de* $G = (V, E)$ *est de la forme*

$$\chi_G(\lambda) = (-\lambda)^n + c_1(-\lambda)^{n-1} + c_2(-\lambda)^{n-2} + \cdots + c_n$$

alors certains coefficients de χ_G *sont en relation directe avec* G *:*

- c_1 *est le nombre de boucles de* G*, en particulier, si* G *est simple,* $c_1 = 0$*.*
- *Si* G *est simple, alors -* c_2 *est le nombre d'arêtes de* G*.*
- *Si* G *est simple, alors* c_3 *est le double du nombre de triangles de* G*.*

Démonstration. Le premier point est immédiat. Le coefficient c_1 est la somme des éléments diagonaux de A_G. Si G est simple, les sous-matrices diagonales de A_G de dimension 2 sont de la forme

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{ou} \quad \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Le coefficient c_2 étant la somme des déterminants de ces sous-matrices ceux-ci valant respectivement -1 et 0, il est clair que $c_2 = -\#E$. Pour le dernier point, on raisonne de la même façon. Les sous-matrices diagonales non nulles de A_G de dimension 3 sont d'une des formes suivantes (à une permutation des lignes et des colonnes près, ce qui ne change pas la valeur du déterminant)

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \text{ou} \quad \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

Les deux premières ont un déterminant nul et la troisième a un déterminant égal à 2. Le coefficient c_3 étant la somme des déterminants de ces sous-matrices et la dernière matrice correspondant à la présence d'un triangle dans G, on obtient le résultat annoncé. □

Remarque 2.5.8 *On voit donc que le polynôme caractéristique de* $A(G)$ *fournit des renseignements sur le graphe* G*. Cependant, deux graphes non isomorphes peuvent avoir le même polynôme caractéristique*[10]*. On parle alors de graphes cospectraux. Par exemple, les deux graphes de la figure 2.37 ont le même spectre. En effet, ils ont tous les deux comme polynôme caractéristique,*

$$-1 + 4\lambda + 7\lambda^2 - 4\lambda^3 - 7\lambda^4 + \lambda^6.$$

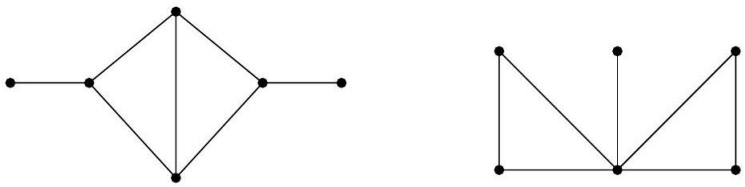


FIGURE 2.37 – Deux graphes cospectraux.

Proposition 2.5.9 *Soit $G = (V, E)$ un graphe biparti. Si λ est valeur propre de G, alors $-\lambda$ l'est aussi. Autrement dit, le spectre d'un graphe biparti est symétrique par rapport à 0.*

Démonstration. Par hypothèse, V se partitionne en deux sous-ensembles V_1 et V_2 de manière telle que toute arête de G est de la forme $\{u, v\}$ avec $u \in V_1$ et $v \in V_2$. Si on ordonne les sommets de V de manière à considérer tout d'abord les sommets de V_1, alors $A(G)$ a la forme

$$\begin{pmatrix} 0 & B \\ \widetilde{B} & 0 \end{pmatrix}$$

où B est une matrice de dimension $\#V_1 \times \#V_2$. Soit x un vecteur propre non nul de $A(G)$ de valeur propre λ. Appelons x_1 (resp. x_2) le vecteur obtenu en considérant les $\#V_1$ premières (resp. les $\#V_2$ dernières) composantes de x.
Ainsi,

$$\begin{pmatrix} 0 & B \\ \widetilde{B} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} Bx_2 \\ \widetilde{B}x_1 \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

Nous pouvons à présent facilement exhiber un vecteur propre non nul de valeur propre $-\lambda$,

$$\begin{pmatrix} 0 & B \\ \widetilde{B} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ -x_2 \end{pmatrix} = \begin{pmatrix} -Bx_2 \\ \widetilde{B}x_1 \end{pmatrix} = \lambda \begin{pmatrix} -x_1 \\ x_2 \end{pmatrix} = -\lambda \begin{pmatrix} x_1 \\ -x_2 \end{pmatrix}.$$

□

Nous montrerons un peu plus loin qu'on dispose également de la réciproque de ce résultat (voir corollaire 2.5.18). On peut aussi définir la matrice d'adjacence d'un multi-graphe orienté. La définition est analogue à celle donnée précédemment sauf qu'on tient ici compte de l'orientation. Il en résulte que la matrice obtenue n'est en général pas symétrique.

Définition 2.5.10 *Soit $G = (V, E)$ un multi-graphe orienté dont les sommets sont ordonnés par $V = \{v_1, \ldots, v_n\}$. La matrice d'adjacence de G est la matrice $A(G)$ dont l'élément $[A(G)]_{i,j}$ est égal au nombre d'arcs (v_i, v_j) présents dans $E, 1 \leq i, j \leq n$.*

Exemple 2.5.11 *Si on considère le graphe de la figure 2.37, on obtient la matrice d'adjacence suivante*

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

10. C'est comme en algèbre matricielle, deux matrices peuvent avoir le même polynôme caractéristique sans pour autant se réduire à la même forme de Jordan.

Théorème 2.5.12 *Soit $G = (V, E)$ un multi-graphe (orienté ou non) tel que $V = \{v_1, \ldots, v_k\}$. Pour tous $i, j \in \{1, \ldots, k\}$ et pour tout $n > 0$,*

$$[A(G)^n]_{i,j}$$

est exactement le nombre de chemins de longueur n joignant v_i à v_j.

Démonstration. On procède par récurrence sur n. Le cas de base $n = 1$ découle de la définition même de la matrice d'adjacence. Supposons le résultat vérifié pour $n > 0$ et vérifions-le pour $n + 1$. On a bien sûr

$$\left[A(G)^{n+1}\right]_{i,j} = \sum_{s=1}^{k} [A(G)^n]_{i,s} [A(G)]_{s,j}$$

Par hypothèse de récurrence, $[A(G)^n]_{i,s}$ compte le nombre de chemins de longueur n joignant v_i à v_s. De plus, $[A(G)]_{s,j}$ compte le nombre d'arcs/arêtes joignant v_s à v_j. Par conséquent, $[A(G)^n]_{i,s} [A(G)]_{s,j}$ compte le nombre de chemins de longueur $n+1$ joignant v_i à v_j en passant par v_s, d'où la conclusion. □

2.5.2 Théorie de Perron-Frobenius

La connexité d'un graphe se traduit par une propriété immédiate de sa matrice d'adjacence. On peut même, dans certains cas, obtenir des renseignements plus fins sur la longueur des chemins joignant deux sommets quelconques d'une composante connexe. Donnons tout d'abord deux définitions concernant les matrices à coefficients positifs ou nuls (faites attention dans les deux énoncés à l'ordre des quantificateurs). Nous verrons ensuite le rapport entre ces matrices et les graphes.

Définition 2.5.13 *Une matrice carrée $A = (a_{ij})_{1 \leq i,j \leq n}$ à coefficients (réels) positifs ou nuls est irréductible si pour tous $i, j \in \{1, \ldots, n\}$, il existe un entier $N(i, j) \geq 0$ tel que*

$$\left[A^{N(i,j)}\right]_{i,j} > 0.$$

Autrement-dit, une matrice irréductible est une matrice dont le graphe associé (orienté, avec un arc $i \to j$ si $a_{ij} > 0$) est fortement connexe. Cela signifie quon peut aller de tout sommet i à tout sommet j par un chemin fini.

Définition 2.5.14 *Une matrice carrée $A = (a_{ij})_{1 \leq i,j \leq n}$ à coefficients (réels) positifs ou nuls est primitive s'il existe un entier $N > 0$ tel que pour tous $i, j \in \{1, \ldots, n\}$*

$$\left[A^N\right]_{i,j} > 0,$$

ce que l'on s'autorise à noter $A^N > 0$ étant sous-entendu que les inégalités sont interprétées composante à composante. On remarque aussi que toute matrice primitive est irréductible.

Toute matrice primitive est irréductible, mais la réciproque est fausse. Une matrice irréductible avec au moins un élément positif sur la diagonale est primitive.
En termes de graphes, une matrice est primitive sl existe un entier N tel que pour tout couple (i, j), il existe un chemin de longueur exactement N reliant i à j. Dans cette sous-section, pour deux matrices réelles A et B de même dimension, il sera commode d'écrire $A < B$ (resp. $\leq, \geq, >$) si l'inégalité a lieu composante à composante. Cela n'entraîne pas d'ambiguïté particulière.

10. Avec un peu d'expérience, le lecteur pourra se convaincre que, dans cette définition, on aurait aussi pu imposer de manière équivalente $N(i, j) > 0$.

Proposition 2.5.15 *Un multi-graphe orienté (resp. non orienté) est fortement connexe (resp. connexe) si et seulement si sa matrice d'adjacence est irréductible.*

Démonstration. C'est une conséquence directe du théorème 2.5.12. □

Remarque 2.5.16 *Si la matrice d'adjacence d'un graphe (orienté ou non) est primitive, cela entraîne que le graphe est non seulement connexe mais qu'il existe N tel que, quelle que soit la paire de sommets considérée, il existe un chemin de longueur N les joignant. Par abus de langage, on s'autorisera alors à parler de graphe primitif.*

Lorsqu'une matrice est irréductible (ou, en particulier primitive), on dispose du puissant théorème de Perron-Frobenius.

Théorème 2.5.17 *(Perron-Frobenius). Soit $A \geq 0$ une matrice carrée irréductible de dimension n.*

- *La matrice A possède un vecteur propre $v_A \in \mathbb{R}^n$ (resp. $w_A \in \mathbb{R}^n$) dont les composantes sont toutes strictement positives et correspondant à une valeur propre $\lambda_A > 0$,*

$$Av_A = \lambda_A v_A \,(\text{resp}\,.\widetilde{w_A}A = \lambda_A\widetilde{w_A})$$

- *Cette valeur propre λ_A possède une multiplicité algébrique (et géométrique) simple.*
- *Tout vecteur propre de A dont les composantes sont strictement positives est un multiple de v_A.*
- *Toute autre valeur propre $\mu \in \mathbb{C}$ de A est telle que $|\mu| \leq \lambda_A$.*
- *Si μ est une valeur propre de A telle que $|\mu| = \lambda_A$, alors*

$$\mu = \lambda_A e^{2ik\pi/d}$$

pour un certain $d \geq 1$ et $k \in \{0, \ldots, d-1\}$. De plus, pour tout $k \in \{0, \ldots, d-1\}, \lambda_A e^{2ik\pi/d}$ est une valeur propre de A.

- *Soit B une matrice réelle à coefficients positifs ou nuls de même dimension que A. Si $B \leq A$, alors pour toute valeur propre μ de B, on a $|\mu| \leq \lambda_A$ et l'égalité a lieu si et seulement si $A = B$.*

La valeur propre λ_A est appelée la valeur propre de Perron de A. Autrement dit, ce théorème stipule qu'une matrice irréductible possède toujours une valeur propre réelle dominante λ_A. Cependant, on peut avoir d'autres valeurs propres de module égal à λ_A mais dans ce cas, celles-ci sont exactement obtenues par multiplication de λ_A par les racines d-ièmes de l'unité.

Démonstration. Puisqu'il s'agit d'un cours de théorie des graphes et pas d'un cours d'algèbre, nous ne donnons pas ici la preuve de ce résultat. Cette preuve n'est pas difficile mais assez longue (elle fait d'ailleurs intervenir des raisonnements de continuité et de passage à la limite). Nous pouvons à présent prouver la réciproque de la proposition 2.5.9.

Corollaire 2.5.18 *Si $G = (V, E)$ est un graphe (non orienté simple) connexe dont le spectre est symétrique par rapport à 0, alors G est biparti.*

10. Remarquons que A étant réel, son polynôme caractéristique χ_A aussi. Si z_0 est racine de χ_A, alors son conjugué $\overline{z_0}$ aussi.

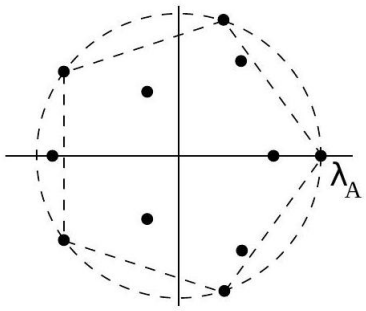


FIGURE 2.38 – Disposition des valeurs propres d'une matrice irréductible.

Démonstration. Soient λ_G la valeur propre de Perron de G et $x \neq 0$ un vecteur propre associé. Par hypothèse $-\lambda_G$ est aussi une valeur propre de G et considérons $y \neq 0$, l'un de ses vecteurs propres. Bien évidemment, x et y sont linéairement indépendants. Si A est la matrice d'adjacence de G, A^2 est la matrice d'adjacence du multi-graphe $G' = (V, E')$ où une arête $\{a, b\}$ appartient à E' si et seulement si il existe $c \in V$ tel que $\{a, c\}$ et $\{b, c\}$ appartiennent à E. On pourrait appeler G', le graphe des chemins de longueur 2 de G.

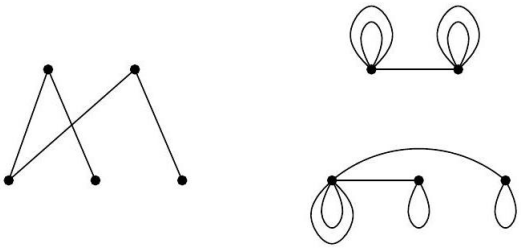


FIGURE 2.39 – Figure II.4. Une illustration des graphes G et G'

Il est clair que λ_G^2 est la valeur propre dominante de A^2 et que x et y en sont des vecteurs propres. Par conséquent, la multiplicité de λ_G^2 est au moins 2 et on en déduit que A^2 ne peut être irréductible (i.e., G' n'est pas connexe). Nous allons montrer que G est biparti. En effet, G' contient au moins deux composantes connexes. En fait, il en contient exactement deux. Comme nous allons l'expliquer, cela découle du fait que G est connexe.
Soit u un sommet quelconque fixé. On définit V_1 (resp. V_2) comme l'ensemble des sommets joints à u par un chemin de longueur impaire (resp. paire) dans G. Puisque G est connexe, $V_1 \cup V_2 = V$ (à ce stade, rien ne garantit qu'il s'agisse d'une partition, on pourrait imaginer qu'un sommet v soit joint à u par deux chemins, l'un de longueur paire, l'autre de longueur impaire). Tous les sommets de V_2 joints à u par un chemin de longueur paire dans G sont connectés à u dans G', cela découle de la définition même de G'. La restriction de G' aux sommets de V_2 est donc connexe. Tous les sommets de V_1 joints à u par un chemin de longueur impaire dans G, sont connectés entre eux dans G'. En effet, considérons $v, w \in V_1$. Dans G, il existe un chemin de longueur impaire entre v_1 et u et aussi un entre u et v_2. Il existe donc un chemin (passant par u) de longueur paire dans G entre v

10. Des vecteurs propres non nuls associés à des valeurs propres distinctes sont linéairement indépendants.
10. Un graphe $G = (V, E)$ est biparti si et seulement si V peut être partitionné en deux sous-ensembles V_1 et V_2 de manière telle que tout chemin joignant un sommet de V_1 à un sommet de V_2 (resp. joignant deux sommets de V_1 ou deux sommets de V_2) soit de longueur impaire (resp. paire).

et w. Autrement dit, la restriction de G' à V_1 est connexe. De plus, $V_1 \cap V_2 = \emptyset$, car sinon G' serait connexe. En raisonnant sur la parité des longueurs de chemins de G, on a donc bien exactement deux composantes connexes V_1 et V_2 dans G'.
Puisque dans G', il n'y a aucune arête entre un sommet de V_1 et un sommet de V_2, on en conclut que dans G, il n'y a aucun chemin de longueur paire entre deux tels sommets (en effet, un chemin de longueur 2ℓ dans G se décompose en ℓ chemins de longueur 2, c'est-à-dire en un chemin de longueur ℓ dans G', ce dernier restant dans une unique composante connexe). Autrement dit, un chemin joignant dans G un sommet de V_1 et un sommet de V_2 est nécessairement de longueur impaire.
Supposons à présent que, dans G, un chemin de longueur impaire $2\ell + 1$ joigne deux sommets u_1 et $u_{2\ell+2}$ de V_1. Par symétrie, le raisonnement qui suit s'applique aussi à deux sommets de V_2. Ce chemin, déterminé par la suite de sommets $(u_1, u_2, \ldots, u_{2\ell+1}, u_{2\ell+2})$, se décompose en ℓ chemins $(u_{2i-1}, u_{2i}, u_{2i+1})$ de longueur $2, i = 1, \ldots, \ell$, plus une arête $\{u_{2\ell}, u_{2\ell+1}\}$. Dans G', les ℓ chemins de longueur 2 correspondent à des arêtes de la composante connexe V_1. Donc $u_1, u_3, \ldots, u_{2\ell+1}$ appartiennent à V_1. On en déduit que dans G, il existe alors une arête entre deux sommets $a = u_{2\ell+1}$ et $b = u_{2\ell+2}$ de V_1. Mais G étant connexe, il existe aussi une arête joignant un sommet c de V_1 et un sommet d de V_2. De là, on en conclut qu'il existe un chemin de longueur paire joignant a à d (en effet, b et c se trouvent dans la même composante connexe de G', ils sont donc séparés par un chemin de longueur paire dans G et il suffit d'ajouter les deux arêtes $\{a, b\}$ et $\{c, d\}$). Ceci est en contradiction avec la première partie de notre raisonnement.
En conclusion, dans G, les sommets de V_1 (resp. de V_2) sont joints entre eux par des chemins de longueur paire exclusivement. En particulier, il n'y a aucune arête entre deux sommets de V_1 (resp. V_2). De plus, les sommets de V_1 sont joints aux sommets de V_2 par des chemins de longueur impaire exclusivement. Ceci montre que G est biparti.
Dans le cas d'une matrice primitive, les assertions du théorème de PerronFrobenius sont encore renforcées (les trois premières étant inchangées). □

Théorème 2.5.19 *(Perron). Soit $A \geq 0$ une matrice carrée primitive de dimension n.*

- *La matrice A possède un vecteur propre $v_A \in \mathbb{R}^n$ (resp. $w_A \in \mathbb{R}^n$) dont les composantes sont toutes strictement positives et correspondant à une valeur propre $\lambda_A > 0$,*

$$Av_A = \lambda_A v_A \,(\text{resp}\,.\widetilde{w_A}A = \lambda_A\widetilde{w_A})$$

- *Cette valeur propre λ_A possède une multiplicité algébrique (et géométrique) simple.*
- *Tout vecteur propre de A dont les composantes sont strictement positives est un multiple de v_A.*
- *Toute autre valeur propre $\mu \in \mathbb{C}$ de A est telle que $|\mu| < \lambda_A$.*
- *Soit B une matrice réelle à coefficients positifs ou nuls de même dimension que A. Si $B \leq A$, alors pour toute valeur propre μ de B, on a $|\mu| \leq \lambda_A$ et l'égalité a lieu si et seulement si $A = B$.*

Ainsi, la valeur propre de Perron λ_A est l'unique valeur propre dominante. Toute autre valeur propre de A a un module strictement inférieur à λ_A. On a donc un résultat plus fort que dans le cas irréductible (c'est assez naturel puisque les hypothèses sont plus fortes).

Traitons à présent deux exemples. Nous y avons choisi des graphes orientés. Des constatations analogues peuvent naturellement être obtenues dans le cas non orienté.

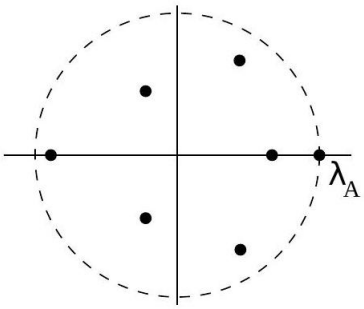
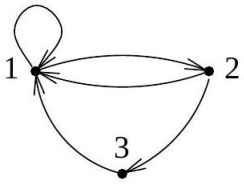


FIGURE 2.40 – Disposition des valeurs propres d'une matrice primitive.



$$A(G) = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

FIGURE 2.41 – Un graphe avec une matrice d'adjacence primitive.

Exemple 2.5.20 *(Cas primitif). Considérons le graphe de la figure 2.41 et la matrice d'adjacence correspondante. La matrice $A(G)$ est primitive. En effet, on a*

$$A(G)^2 = \begin{pmatrix} 2 & 1 & 1 \\ 2 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} \quad et \quad A(G)^3 = \begin{pmatrix} 4 & 2 & 1 \\ 3 & 2 & 1 \\ 2 & 1 & 1 \end{pmatrix} > 0.$$

Il existe donc un chemin de longueur 3 joignant toute paire de sommets.
On a les chemins suivants :

De 1 à 1 (4 chemins) :

1. $1 \to 1 \to 1 \to 1$
2. $1 \to 1 \to 2 \to 1$
3. $1 \to 2 \to 1 \to 1$
4. $1 \to 2 \to 3 \to 1$

De 1 à 2 (2 chemins) :

1. $1 \to 1 \to 1 \to 2$
2. $1 \to 2 \to 1 \to 2$

De 1 à 3 (1 chemin) :

1. $1 \to 1 \to 2 \to 3$

De 2 à 1 (3 chemins) :

1. $2 \to 1 \to 1 \to 1$
2. $2 \to 1 \to 2 \to 1$
3. $2 \to 3 \to 1 \to 1$

De 2 à 2 (2 chemins) :

1. $2 \to 1 \to 1 \to 2$
2. $2 \to 3 \to 1 \to 2$

De 2 à 3 (1 chemin) :

1. $2 \to 1 \to 2 \to 3$

De 3 à 1 (2 chemins) :

1. $3 \to 1 \to 1 \to 1$
2. $3 \to 1 \to 2 \to 1$

De 3 à 2 (1 chemin) :

1. $3 \to 1 \to 1 \to 2$

De 3 à 3 (1 chemin) :

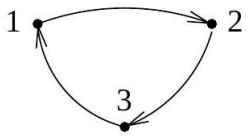
1. $3 \to 1 \to 2 \to 3$

Des valeurs approchées de valeurs propres de $A(G)$ sont

$$\lambda_A \simeq 1.83929, \quad \lambda_2 \simeq -0.41964 + 0.60629i \quad \text{et} \quad \lambda_3 \simeq -0.41964 - 0.60629i$$

et $|\lambda_2| = |\lambda_3| < \lambda_A$.

Exemple 2.5.21 *(Cas irréductible). Considérons à présent le graphe de la figure 2.42 et la matrice d'adjacence correspondante. Il est clair que le graphe est f. connexe et donc, la matrice $A(G)$ est au moins irréductible. Cependant, un chemin de longueur k joint les sommets 1 et 2 si et seulement si un chemin de longueur $k+1$ joint les sommets 1 et 3. Il n'existe donc pas d'entier n pour lequel tout sommet peut être joint à tout autre sommet par un chemin de longueur n. La matrice $A(G)$ n'est donc pas primitive. On pourrait aussi s'en convaincre en montrant que, pour tout n,*



$$A(G) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

FIGURE 2.42 – Un graphe avec une matrice d'adjacence irréductible.

$$A(G)^{3n} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, A(G)^{3n+1} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, A(G)^{3n+2} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Les valeurs propres sont ici les racines cubiques de l'unité

$$1, e^{2i\pi/3}, e^{4i\pi/3}$$

et donc, on a ici, par opposition avec l'exemple précédent, plusieurs valeurs propres de module maximum (=1).

Il est aussi intéressant de noter que pour joindre deux sommets fixés, uniquement certaines longueurs de chemin peuvent être considérées. Par exemple, pour joindre les sommets 2 et 3, uniquement des chemins de longueur congrue à 1 modulo 3 peuvent être envisagés. Ce phénomène est en fait tout à fait général et sera explicité à la section suivante.

Corollaire 2.5.22 *Soit $A \geq 0$ une matrice carrée. Les assertions suivantes sont équivalentes.*
i) A est primitive,
ii) il existe $N \geq 1$ tel que $A^N > 0$,
iii) il existe $N \geq 1$ tel que $A^n > 0$ pour tout $n \geq N$.

Démonstration. Par définition, i) $\Rightarrow$ ii) et ii) $\Rightarrow$ i). Montrons que ii) $\Rightarrow$ iii). Puisque $A^N > 0$, on en déduit que toute colonne de A contient au moins un élément strictement positif. Par conséquent, si $A^k > 0$, alors $A^k \cdot A > 0$ et de proche en proche, $A^{k+i} > 0$ pour tout $i \geq 0$. Enfin, il est immédiat que iii) $\Rightarrow$ ii). □

2.5.2.1 Période d'une matrice irréductible

Soit $(a_{ij})_{1\leq i,j\leq d} = A \geq 0$ une matrice carrée de dimension d à coefficients positifs ou nuls. Par indice de A, on entend un élément de $\{1, \ldots, d\}$.

Définition 2.5.23 *Soit i un indice. S'il existe $N > 0$ tel que $\left[A^N\right]_{i,i} > 0$, alors la période de l'indice i est le p.g.c.d. de l'ensemble des entiers $n > 0$ pour lesquels*

$$[A^n]_{i,i} > 0.$$

On la note $p(i)$. Bien évidemment, le p.g.c.d. d'un ensemble infini d'entiers $X = \{x_1 < x_2 < \cdots\} \subseteq \mathbb{N}$ est le plus grand entier p appartenant à l'ensemble fini $\{1, 2, \ldots, x_1\}$ tel que pour tout $k \geq 1, p$ divise x_k.

Remarque 2.5.24 *Cette définition est donnée en termes de matrices, mais elle possède un analogue immédiat en termes de graphes. En effet, à la matrice $A = (a_{ij})_{1\leq i,j\leq d}$, on fait correspondre un graphe $G_A = (V_A, E_A)$ ayant pour ensemble de sommets $V_A = \{1, \ldots, d\}$ et il existe un arc joignant i à j si et seulement si $a_{i,j} > 0$. Ainsi, pour définir la période d'un sommet $i \in V_A$ appartenant à une composante f. connexe du graphe G_A, on recherche le p.g.c.d. de l'ensemble des entiers k pour lesquels il existe au moins un circuit de longueur k passant par i.*

Exemple 2.5.25 *Dans l'exemple 2.5.20, on a $[A(G)^n]_{3,3} > 0$ pour tout $n \geq 3$ et le p.g.c.d. des éléments de l'ensemble $X = \{3, 4, 5, \ldots\}$ est 1. Ainsi, le sommet 3 est de période 1.*

Par contre, dans l'exemple 2.5.21, $[A(G)^n]_{i,i} > 0$ si et seulement si n est un multiple de 3. Ainsi, les périodes des sommets de ce graphe sont toutes égales à 3.

Lemme 2.5.26 *Soient i, j deux indices de $A \geq 0$. S'il existe m, n tels que $[A^m]_{i,j} > 0$ et $[A^n]_{j,i} > 0$, alors $p(i) = p(j)$.*

Remarque 2.5.27 *En utilisant la remarque 2.5.24, le lemme se réexprime comme suit. S'il existe dans G_A un chemin de longueur m joignant i et j et un chemin de longueur n joignant j et i, autrement dit si $i \leftrightarrow j$, alors les deux sommets ont même période. (Ou encore, tous les sommets d'une composante f. connexe ont même période.)*

Démonstration. Pour tout s tel que $[A^s]_{j,j} > 0$, on a

$$\begin{aligned}\left[A^{m+s+n}\right]_{i,i} &= \sum_{k=1}^{d} \left[A^{m+s}\right]_{i,k} [A^n]_{k,i} \\ &\geq \left[A^{m+s}\right]_{i,j} [A^n]_{j,i} \\ &= \sum_{k=1}^{d} [A^m]_{i,k} [A^s]_{k,j} [A^n]_{j,i} \\ &\geq [A^m]_{i,j} [A^s]_{j,j} [A^n]_{j,i} > 0.\end{aligned}$$

Les deux inégalités proviennent du fait que les éléments de A sont positifs ou nuls. Pour un tel s, on a aussi $[A^{2s}]_{j,j} > 0$ (en effet, si on effectue le produit matriciel $[A^s.A^s]_{j,j}$, on retrouve le terme $[A^s]_{j,j} \cdot [A^s]_{j,j}$ et les autres termes de la somme sont positifs ou nuls). Dès lors, on a aussi

$$\left[A^{m+2s+n}\right]_{i,i} > 0.$$

Par conséquent, $p(i)$ divise $m + 2s + n$ et $m + s + n$ et aussi leur différence, s. En conclusion, pour tout s tel que $[A^s]_{j,j} > 0, p(i)$ divise s et donc $p(i) \leq p(j)$. Par symétrie, on a aussi que $p(j) \leq p(i)$ et donc $p(i) = p(j)$. □

Grâce à ce lemme, nous pouvons introduire la définition suivante. En effet, pour une matrice irréductible, toutes les périodes sont nécessairement identiques. En particulier, tous les sommets d'une composante f. connexe de G_A ont même période.

Définition 2.5.28 *Une matrice irréductible $A \in \mathbb{R}_d^d$ est cyclique de période p si tous les indices de A sont de période $p > 1$. Sinon, tous les indices sont de période $p = 1$ et A est dite acyclique.*

Lemme 2.5.29 *Soit $A \geq 0$ une matrice carrée irréductible de période $p \geq 1$. Soit i un indice de A. Il existe $N_i \geq 0$ tel que pour tout $n \geq N_i$, $[A^{np}]_{i,i} > 0$.*

Démonstration. Supposons tout d'abord que $\left[A^{kp}\right]_{i,i} > 0$ et $\left[A^{\ell p}\right]_{i,i} > 0$. Dès lors (même raisonnement que dans la preuve du lemme 2.5.26),

$$\left[A^{(k+\ell)p}\right]_{i,i} \geq \left[A^{kp}\right]_{i,i} \left[A^{\ell p}\right]_{i,i} > 0.$$

Cela signifie que l'ensemble $\mathcal{S}$ des multiples np de p qui sont tels que $[A^{np}]_{i,i} > 0$ est stable pour l'addition (et $\mathcal{S}$ contient au moins un multiple de p). De plus, par définition, le p.g.c.d. des éléments de $\mathcal{S}$ vaut p. La conclusion découle alors du lemme suivant. □

Lemme 2.5.30 *Soit $X \subseteq \mathbb{N}$ un ensemble d'entiers stable pour l'addition. Alors X contient tous les multiples du p.g.c.d. des éléments de X à l'exception éventuellement d'un nombre fini d'entre eux.*

Démonstration. Soit p le p.g.c.d. des éléments de X. Quitte à diviser les éléments de X par p, on peut supposer que $p = 1$. Dès lors, il existe un ensemble fini [11] $\{x_1, \ldots, x_k\} \subseteq X$ tel que

10. $p(i)$ est un diviseur commun des éléments de l'ensemble $\left\{s > 0 \mid [A^s]_{j,j} > 0\right\}$ et d'autre part, $p(j)$ le p.g.c.d. de cet ensemble.

11. Nous savons que le p.g.c.d. de $X = \{x_1 < x_2 < \cdots\}$ vaut 1 . Si on considère tout d'abord X restreint à $\{x_1\}$ seul, le p.g.c.d. potentiel serait x_1. Puis, quand on considère $\{x_1, x_2\}$, le p.g.c.d. potentiel ne peut que diminuer (ou au mieux rester constant) puisqu'on doit considérer cette fois les facteurs premiers communs à x_1 et à x_2. A l'étape suivante, on considère $\{x_1, x_2, x_3\}$ et ainsi de suite (à chaque étape, le p.g.c.d. décroît). Nous affirmons qu'il existe k tel que le p.g.c.d. de $\{x_1, \ldots, x_k\}$ soit 1 car si tel n'était pas le cas, le p.g.c.d. de X serait > 1, ce qui est contraire à notre supposition. Remarquons en particulier que k peut être > 2, ne serait-ce par exemple qu'avec l'ensemble fini $\{6, 10, 15\}$ dont le p.g.c.d. vaut 1 mais qui est tel que ses éléments pris deux à deux ne sont pas premiers entre eux.

$$\text{p.g.c.d. } \{x_1, \ldots, x_k\} = 1$$

Par le théorème de Bezout, il existe des entiers relatifs $\lambda_1, \ldots, \lambda_k \in \mathbb{Z}$ tels que

$$\lambda_1 x_1 + \cdots + \lambda_k x_k = 1.$$

Si on regroupe tous les termes dont les coefficients λ_i sont positifs (resp. négatifs), cette somme se réécrit

$$m - n = 1$$

avec $m, n \in X$ car X est stable pour l'addition. Soit q un entier tel que $q \geq n(n-1)$. Par division euclidienne,

$$q = an + b, \quad 0 \leq b < n$$

De plus, $a \geq n - 1$. Puisque $m - n = 1$, il vient

$$q = an + b(m - n) = (a - b)n + bm$$

avec $a - b \geq 0$. On en conclut que q appartient à X (car $m, n \in X$). Nous avons donc montré que tout entier $q \geq n(n-1)$ appartient à X. Cela termine la preuve. □

Le résultat suivant montre que l'exemple 2.5.21 est l'archétype même de la situation rencontrée dans le cas d'un graphe irréductible.

Théorème 2.5.31 *Soit $A \geq 0$ une matrice carrée irréductible de période $p \geq 1$. Pour toute paire i, j d'indices de A, il existe un unique entier $r_{i,j} \in \{0, \ldots, p-1\}$ tel que*

- *$[A^n]_{i,j} > 0$ entraîne $n \equiv r_{i,j} (\mathrm{mod} p)$ et*
- *il existe $N_{i,j}$ tel que $[A^{np+r_{i,j}}]_{i,j} > 0$ pour tout $n \geq N_{i,j}$.*

.

Exemple 2.5.32 *On peut reprendre l'exemple 2.5.21. Dans cet exemple, $p = 3$ et si on fixe le sommet $i = 2$, on a $r_{2,1} = \mathbf{2}, r_{2,2} = \mathbf{0}$ et $r_{2,3} = \mathbf{1}$. En effet, $[A(G)^{3n+\mathbf{2}}]_{2,1} = 1, [A(G)^{3n+\mathbf{0}}]_{2,2} = 1$ et $[A(G)^{3n+\mathbf{1}}]_{2,3} = 1$.*

Démonstration. Supposons que $[A^m]_{i,j} > 0$ et $[A^n]_{i,j} > 0$. Nous allons montrer que $m \equiv n (\mathrm{mod} p)$. Puisque A est irréductible, il existe ℓ tel que $\left[A^\ell\right]_{j,i} > 0$. Dès lors,

$$\left[A^{m+\ell}\right]_{i,i} \geq [A^m]_{i,j} \left[A^\ell\right]_{j,i} > 0 \quad \text{et} \quad \left[A^{n+\ell}\right]_{i,i} > 0.$$

La période p divise donc $m + \ell$ et $n + \ell$ donc leur différence. Autrement dit, $m - n \equiv 0 (\mathrm{mod} p)$. Passons à la deuxième partie. Puisque A est irréductible, il existe ℓ tel que $\left[A^\ell\right]_{i,j} > 0$ et au vu de la première partie,

$$\ell = mp + r_{i,j}$$

Posons

$$N_{i,j} = N_i + m$$

(avec N_i donné dans le lemme 2.5.29). Par définition de N_i, on a

$$\forall n \geq N_i, [A^{np}]_{i,i} > 0$$

De là, si $k \geq N_{i,j}$, alors

$$kp + r_{i,j} = (n+m)p + r_{i,j} \quad \text{avec } n \geq N_i$$

et

$$\left[A^{kp+r_{i,j}}\right]_{i,j} \geq \left[A^{np}\right]_{i,i} \left[A^{mp+r_{i,j}}\right]_{i,j} > 0.$$

□

Proposition 2.5.33 *Une matrice irréductible est acyclique si et seulement si elle est primitive.*

Démonstration. Si la matrice est acyclique (i.e., de période $p = 1$), alors avec les notations du théorème 2.5.31, $r_{i,j} = 0$ quels que soient les indices i et j. On en conclut que

$$\left[A^n\right]_{i,j} > 0 \quad \text{si } n \geq N_{i,j}$$

Ainsi, si on pose $\mathcal{N} = \sup_{i,j} N_{i,j}$, alors $A^{\mathcal{N}} > 0$ et A d'être primitive.
Réciproquement si A est primitive, A est nécessairement irréductible et pour k suffisamment grand et pour tout indice i de A, $\left[A^k\right]_{i,i} > 0$ et $\left[A^{k+1}\right]_{i,i} > 0$. Le p.g.c.d. de k et de $k+1$ étant 1, la conclusion en découle. □

Exemple 2.5.34 *Terminons par une application des résultats précédents et considérons les graphes représentés à la figure 2.43. Le sommet A appartient visiblement à un cycle de longueur 1 (une boucle). Par conséquent, sa période vaut 1 et le graphe est primitif au vu de la proposition précédente. Le sommet B appartient quant à lui à un cycle de longueur 2 et à un cycle de longueur 3. Ainsi, le p.g.c.d. des longueurs des cycles passant par B vaut 1 et le graphe est encore une fois primitif. En particulier, cela signifie que pour tout n suffisamment grand, il existe un chemin de longueur n entre tout couple de sommets. Enfin, tout cycle contenant C est de longueur $4p+2q$, $p, q \geq 0$. On en conclut que la période des différents sommets de ce dernier graphe vaut 2. Pour terminer cet exemple, notons encore que les chemins joignant D à E sont de longueur $1, 5, 7, 9, \ldots$ ainsi on dispose d'un chemin entre D et E si et seulement si sa longueur est de la forme $1+2n$ avec $n \geq 2$ ce qui illustre parfaitement la seconde partie du théorème* **??**

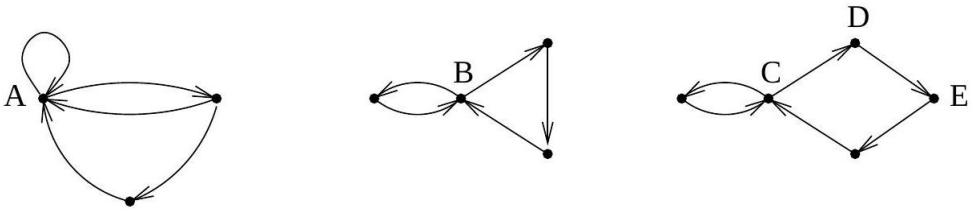


FIGURE 2.43 – Trois graphes f. connexes.

Signalons sans démonstration un dernier résultat, sorte de réciproque au théorème de Perron-Frobenius.

Proposition 2.5.35 *Si $A \geq 0$ est une matrice irréductible possédant une valeur propre dominante λ (i.e., pour toute valeur propre $\mu \neq \lambda$ de A, $|\mu| < \lambda$), alors A est primitive.*

2.5.2.2 Estimation du nombre de chemins de longueur n

Le théorème de Perron permet de donner le comportement asymptotique du nombre de chemins de longueur n joignant deux sommets quelconques d'un graphe dont la matrice d'adjacence est primitive. Avec les notations du théorème de Perron, si A est une matrice primitive, alors il est possible de montrer que

$$A^k = \lambda_A^k v_A \widetilde{w_A} + o\left(\lambda_A^k\right) \tag{3}$$

où v_A et $\widetilde{w_A}$ sont des vecteurs propres choisis de telle sorte que $\widetilde{w_A} \cdot v_A = 1$. Remarquez que $v_A\widetilde{w_A}$ est une matrice carrée de dimension n dont les éléments sont tous strictement positifs. Autrement dit, ce résultat stipule qu'asymptotiquement, tout élément de A^k est proportionnel à λ_A^k (la constante de proportionnalité dépendant de l'élément envisagé). Tout comme nous avons admis le théorème de Perron, nous admettrons aussi ce résultat sortant du cadre que nous nous sommes fixés dans ce cours.

Remarque 2.5.36 *Il est même possible d'obtenir des développements plus fins du terme d'erreur en l'exprimant à l'aide de la deuxième valeur propre (par module décroissant) de A.*

Exemple 2.5.37 *Si on poursuit l'exemple 2.5.20, pour tout couple $(i,j) \in \{1,2,3\} \times \{1,2,3\}$, il existe une constante $d_{i,j} > 0$ telle que le nombre $c_{i,j}(n)$ de chemins de longueur n joignant i à j satisfasse*

$$\lim_{n\to\infty} \frac{c_{i,j}(n)}{d_{i,j}\lambda_A^n} = 1.$$

Remarque 2.5.38 *Dans le cas d'un graphe f. connexe qui n'est pas primitif, on dispose du théorème de Perron-Frobenius. Néanmoins, si on a plusieurs valeurs propres de module maximum, des compensations entre celles-ci peuvent se produire, et fournir une estimation des $c_{i,j}(n)$ semblable à celle donnée ci-dessus n'est pas si simple. En effet, si on reprend une fois encore l'exemple 2.5.21 et que l'on s'intéresse à la suite formée des nombres de chemins de longueur n joignant 1 à 3 pour $n = 0, 1, 2, \ldots$, on obtient*

$$0, 0, 1, 0, 0, 1, 0, 0, 1, \ldots$$

qui est clairement une suite divergente. Ainsi, la limite

$$\lim_{n\to\infty} \frac{c_{1,3}(n)}{\lambda_A^n}$$

n'existe pas ! Ceci s'explique par le fait que des combinaisons convenables de puissances des racines de l'unité s'annulent [12] *:*

$$\frac{\left(e^{2i\pi/3}\right)^n + \left(e^{4i\pi/3}\right)^n + 1}{3} = 0, \ \textit{si } n \equiv 1, 2 \pmod 3.$$

11. Autrement dit, quand $n \to +\infty$.

11. Pour rappel, une fonction $f(x)$ est en $o(g)$ si f/g tend vers 0 si $x \to \infty$. Cela se lit " f est en petit oh de g ".

[12] Par un résultat classique d'algèbre linéaire (conséquence de la théorie de la diagonalisation ou de la mise sous forme normale de Jordan), si $\lambda_1, \ldots, \lambda_p$ sont les valeurs propres distinctes d'une matrice M et sont zéros de son polynôme minimum de multiplicité $m_1, \ldots, m_p$, alors pour tout entier k,

$$\left(M^k\right)_{i,j} = \sum_{t=1}^{p} P_{i,j}^{(t)} \lambda_t^k$$

2.5.2.3 Cas d'un graphe ayant plusieurs composantes f. connexes.

Nous pouvons obtenir des résultats plus fins que ceux obtenus ci-dessus. On considère le condensé $\mathcal{C}$ d'un graphe G (ou graphe acyclique des composantes cf. définition **??**) dont les sommets sont les composantes f. connexes de G. Puisque le condensé est sans cycle, on peut ordonner ses sommets par tri topologique. Supposons disposer d'un tel ordonnancement. La matrice d'adjacence de $\mathcal{C}$ a alors une forme triangulaire supérieure et si on ordonne les éléments de G en considérant les sommets des composantes f. connexes prises une à une en respectant le tri topologique, la matrice d'adjacence de G est alors une matrice bloc triangulaire supérieure.

Exemple 2.5.39 *Poursuivons l'exemple 2.2.26. Pour celui-ci, on obtient la matrice suivante si on considère l'ordre donné à la figure 2.28.*

$$A(G) = \left(\begin{array}{ccc|cccccccccc} 0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0\\ 1&0&0&1&0&0&0&1&0&0&0&0\\ \hline 0&0&0&0&1&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&1&1&0&0 \end{array}\right).$$

Remarque 2.5.40 *Au vu de l'exemple précédent, il est immédiat d'observer que le spectre d'un graphe est l'union des spectres de ses composantes connexes.*

Nous nous restreignons une fois encore au cas où les composantes $f.$ connexes sont des sous-graphes primitifs.
Soit un graphe fini possèdant deux composantes f . connexes primitives A et B et deux sommets $a \in A$ et $b \in B$ tels que $a \to b$. Les chemins joignant un sommet de A à un sommet de B sont en nombre fini et de longueur bornée. Si λ_A et λ_B sont les valeurs propres de Perron de A et de B respectivement, on en déduit que le nombre $c_{a,b}(n)$ de chemins de longueur n joignant a à b est proportionnel [12] à

$$\sum_{i=0}^{n} \lambda_A^i \lambda_B^{n-i} = \lambda_B^n \sum_{i=0}^{n} \left(\frac{\lambda_A}{\lambda_B}\right)^i$$

où $P_{i,j}^{(t)}$ est un polynôme de degré $< m_t$. Ainsi, lorsqu'on considère un quotient comme $\left(M^k\right)_{i,j} / \lambda_M^k$, ne subsistent à la limite que les valeurs propres de module maximum. Dans l'exemple 2.5.21, n'apparaissent que les trois racines de l'unité.

— Si $\lambda_A = \lambda_B$, alors [14]

$$c_{a,b}(n) \asymp n\lambda^n$$

12. Dans une estimation asymptotique, c'est le comportement général qui est intéressant. La détermination précise de la constante multiplicative nous importe peu.

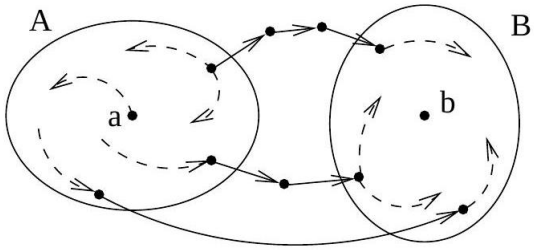


FIGURE 2.44 – Deux composantes f. connexes.

— Sinon, $\lambda_A \neq \lambda_B$ et

$$\sum_{i=0}^{n} \lambda_A^i \lambda_B^{n-i} = \frac{\lambda_A^{n+1} - \lambda_B^{n+1}}{\lambda_A - \lambda_B} \asymp [\max(\lambda_A, \lambda_B)]^n$$

Nous détaillons à présent le cas d'un graphe fini ayant trois composantes f. connexes A, B et C. Nous désirons estimer le nombre $c_{a,B,c}(n)$ de chemins de longueur n joignant $a \in A$ à $c \in C$ en passant par un sommet quelconque de B. Le lecteur pourra aisément adapter les raisonnements au cas général. Avec les mêmes notations que ci-dessus, puisque le nombre de sommets de B est fini, le nombre de

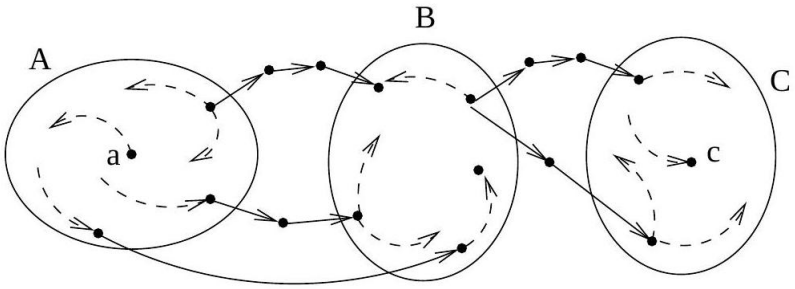


FIGURE 2.45 – Trois composantes f. connexes.

chemins recherché est proportionnel à

$$\sum_{i=0}^{n} \sum_{j=0}^{n-i} \lambda_A^i \lambda_B^j \lambda_C^{n-i-j} \tag{2.3}$$

Nous traitons trois cas.

— Si $\lambda_A = \lambda_B = \lambda_C$, alors (2.3) devient

$$\lambda_A^n \sum_{i=0}^{n} (n+1-i) = \lambda_A^n \left[(n+1)^2 - \frac{n(n+1)}{2} \right] \asymp n^2 \lambda_A^n.$$

— Si $\lambda_A = \lambda_B \neq \lambda_C$ (par symétrie, les autres cas se traitent de la même façon), alors (2.3) devient

$$\sum_{i=0}^{n} \lambda_A^i \underbrace{\sum_{j=0}^{n-i} \lambda_A^j \lambda_C^{n-i-j}}_{\frac{\lambda_A^{n-i+1} - \lambda_C^{n-i+1}}{\lambda_A - \lambda_C}} = \frac{(n+1)\lambda_A^{n+1}}{\lambda_A - \lambda_C} - \frac{\lambda_C}{\lambda_A - \lambda_C} \underbrace{\sum_{i=0}^{n} \lambda_A^i \lambda_C^{n-i}}_{\frac{\lambda_A^{n+1} - \lambda_C^{n+1}}{\lambda_A - \lambda_C}}$$

Ainsi, si $\lambda_A = \lambda_B > \lambda_C$, on trouve

$$c_{a,B,c}(n) \asymp n\lambda_A^n$$

et si $\lambda_A = \lambda_B < \lambda_C$, on trouve

$$c_{a,B,c}(n) \asymp \lambda_C^n$$

— Enfin, si λ_A, λ_B et λ_C sont 2 à 2 distincts, alors (2.3) devient

$$\sum_{i=0}^{n} \lambda_A^i \underbrace{\sum_{j=0}^{n-i} \lambda_B^j \lambda_C^{n-i-j}}_{\frac{\lambda_B^{n-i+1}-\lambda_C^{n-i+1}}{\lambda_B-\lambda_C}} = \frac{1}{\lambda_B - \lambda_C}\left(\lambda_B \sum_{i=0}^{n} \lambda_A^i \lambda_B^{n-i} - \lambda_C \sum_{i=0}^{n} \lambda_A^i \lambda_C^{n-i}\right)$$

et on obtient

$$c_{a,B,c}(n) \asymp \left[\max\left(\lambda_A, \lambda_B, \lambda_C\right)\right]^n.$$

En conclusion, il suffit de détecter la plus grande valeur de Perron λ des différentes composantes connexes par lesquelles passent les chemins d'intérêt et de compter le nombre k de composantes ayant cette valeur propre comme valeur dominante. Le nombre de chemins de longueur n se comporte alors asymptotiquement comme $n^{k-1}\lambda^n$. Le lecteur adaptera facilement les développements effectués ci-dessus au cas d'un nombre arbitraire de composantes primitives. On pourrait aussi regarder la forme de Jordan de la matrice d'adjacence associée, un bloc de taille k associé à la valeur propre λ fait intervenir un polynôme de degré $k-1$ multiplié par une exponentielle λ^n.

2.5.3 Une application : l'algorithme de PageRank

Dans cette section, nous allons illustrer certainement l'une des applications les plus célèbres du théorème de Perron-Frobenius : le succès de Google basé sur leur algorithme de classement des pages web ! Volontairement, plusieurs aspects ne sont pas pris en compte dans cette présentation (développement et efficacité des méthodes numériques, choix de la constante α, mise en pratique, sensibilité aux perturbations, modèle plus évolué, ...). Pour plus de détails, consulter par exemple l'excellent ouvrage de C. Meyer et A. Langville consacré entièrement à ce sujet.
Lorsqu'on effectue une recherche sur un mot clé donné, Google trie les pages contenant ce mot clé en se basant (notamment) sur une mesure, appelée "PageRank", destinée à quantifier la qualité des pages et à déterminer si elles font ou non autorité dans le domaine envisagé. L'obtention d'un "bon" classement des pages sur un sujet donné est d'ailleurs l'une des raisons pour lesquelles Google est si populaire. L'idée originale de L. Page et S. Brin [13] est très simple :

— on accorde plus d'importance, i.e., un score de "PageRank" plus élevé, aux pages référencées par des pages qui font elles-mêmes autorité dans le domaine, c'est-à-dire qui ont un PageRank élevé ;

12. Nous ne parlons pas ici des problèmes d'indexation de pages web, de la recherche par mots clés, etc.... Nous nous concentrons uniquement sur le classement des pages sélectionnées comme apparentées à la recherche. Cette méthode de classement a d'ailleurs été appliquée à d'autres situations (championnats sportifs par exemple)

13. Pour l'anecdote, il faut savoir que le père de Sergeï Brin, Michael Brin est un mathématicien spécialiste des systèmes dynamiques.

— on accorde d'autant moins de crédit à une citation si elle provient d'une page qui dispose de nombreux liens (on ne peut qu'accorder moins de poids aux sites qui galvaudent leurs recommandations).
On suppose disposer d'un graphe simple $G = (V, E)$ où les sommets notés $1, \ldots, n$ représentent les pages de l'Internet (en 2005, on recensait plus de huit milliards de pages) et où l'on dispose d'un arc (i, j) si et seulement si la page i possède un lien pointant vers la page j. Au vu du modèle proposé par Page et Brin, le PageRank $\pi_j \geq 0$ de la page $j \in \{1, \ldots, n\}$ serait donné par

$$\pi_j = \sum_{i \in \text{pred}(j)} \frac{\pi_i}{d^+(i)} \tag{2.4}$$

qui est une formule récursive pour laquelle on ne dispose pas a priori de méthode permettant d'assurer l'existence, l'unicité ou le calcul efficace d'une solution $\pi = (\pi_1, \ldots, \pi_n)$ non triviale. On peut de plus supposer que les scores recherchés sont normalisés,

$$\sum_{i=1}^{n} \pi_i = 1.$$

Le problème peut se réécrire sous forme matricielle ("H" comme "hyperlien"),

$$\pi = \pi H \tag{2.5}$$

où

$$H_{ij} = \begin{cases} A(G)_{ij}/d^+(i) & \text{si } d^+(i) > 0 \\ 0 & \text{si } d^+(i) = 0, \end{cases}$$

avec $A(G)$ la matrice d'adjacence du graphe G et rappelons que π est un vecteur ligne. Sous cette forme, il n'est pas clair que le problème possède une solution (ni qu'elle soit unique et on dispose encore moins d'une méthode de calcul). Autrement dit, il n'y a pas de raison évidente pour laquelle

α	nombre d'itérations
0,5	34
0,75	81
0,8	104
0,85	142
0,9	219
0,95	449
0,99	2292
0,999	23015

TABLE 2.3 – Rôle du paramètre α

1 est valeur propre de H et qui plus est, valeur propre simple. L'astuce pour pouvoir appliquer la théorie de Perron-Frobenius est double. Tout d'abord, pour se débarrasser des "puits", i.e., des pages ne pointant vers aucune autre page et pour obtenir une matrice stochastique, on introduit une matrice S ("S" comme "stochastique") définie par

$$S_{ij} = \begin{cases} A(G)_{ij}/d^+(i) & \text{si } d^+(i) > 0 \\ 1/n & \text{si } d^+(i) = 0. \end{cases}$$

13. Et dans l'ensemble des liens de l'Internet, il y a de nombreux puits ; imaginez tous les liens pointant vers un fichier (image, vidéo, pdf, etc...)

Ensuite, pour assurer la forte connexité du graphe (i.e., le caractère irréductible), on construit une matrice G ("G" comme Google) donnée par la combinaison affine (et même convexe) suivante avec un réel $\alpha \in [0,1]$ fixé,

$$G = \alpha S + (1-\alpha)J/n$$

où $J = (1)_{1\leq i,j\leq n}$. L'équation initiale (6) est remplacée par

$$\pi = \pi G.$$

G est un point du segment $[G, J/n]$ de l'espace des matrices.
(La matrice J/n est parfois appelée matrice de téléportation, cf. remarque suivante.) Le lecteur préférant manipuler des vecteurs propres à droite écrira $\widetilde{\pi} = \widetilde{G}\widetilde{\pi}$. Google attribue à α une valeur de 0,85. Ce choix n'est pas arbitraire. Au plus α est proche de 1, au mieux on approche le modèle "naturel" proposé initialement : on diminue le rôle artificiel de la matrice de téléportation. Cependant, on peut montrer que ce paramètre α contrôle la vitesse de convergence de la méthode de calcul développée et donc le nombre d'itérations à effectuer pour obtenir une estimation du vecteur π (par calcul de puissances successives de G qui contient plus de 8.10^9 entrées). Quand α tend vers 1, ce nombre devient prohibitif comme le montre la table 2.3. Ainsi, $\alpha = 0{,}85$ semble un bon compromis entre le caractère artificiel introduit par la matrice de téléportation et la masse de calculs à réaliser. De plus, on peut montrer par une discussion plus fine sur les valeurs propres des matrices envisagées qu'au plus α est proche de 1, au plus le vecteur π est sensible aux petites pertubations de la matrice H (ce qui s'avère être gênant vu la grande volatilité du web et de sa structure, de nombreux liens apparaissent et disparaissent chaque jour). Idéalement, on désirerait obtenir une mesure peu sensible à de telles pertubations.

Remarque 2.5.41 *On peut donner une interprétation "probabiliste" de la matrice G. Considérons un surfeur qui, se trouvant sur une page quelconque, a deux choix possibles. Soit, avec une probabilité α, il clique au hasard et de manière uniforme sur l'un des liens de la page pour changer de page. Soit, avec une probabilité $1-\alpha$, il choisit au hasard et de manière uniforme l'une des n pages de l'Internet tout entier. Ainsi, G_{ij} représente la probabilité de transition lorsque le surfeur se trouve sur la page i de passer à la page j.*

Ainsi, partant d'une distribution initiale de probabilités, par exemple $\pi^{(0)} = \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$, l'application de G^k permet d'estimer la probabilité de notre surfeur de se trouver sur l'une des pages $1, \ldots, n$ après k clics,

$$\pi^{(k)} = \pi^{(0)} G^k.$$

Nous allons à présent montrer que l'utilisation de cette matrice G (à la place de H) assure l'existence et l'unicité d'une distribution limite π.

Remarque 2.5.42 *Par rapport à l'équation initiale (2.4), l'emploi de la matrice G donne la formule suivante pour la détermination des "nouveaux" π_j qui seront effectivement calculés. On utilise la définition de S_{ij} pour obtenir*

$$\pi_j = \sum_{i=1}^{n} \pi_i \left(\alpha S_{ij} + (1-\alpha)\frac{1}{n}\right)$$

$$= \alpha \sum_{i\in \mathrm{pred}(j)} \frac{\pi_i}{d^+(i)} + \frac{1}{n}\left(1 - \alpha + \alpha \sum_{i:d^+(i)=0} \pi_i\right).$$

Nous nous sommes donc éloignés quelque peu du modèle initialement proposé mais nous allons voir que ces modifications vont permettre un calcul efficace (en assurant de plus l'existence et l'unicité d'une solution !). On observe que le premier terme, à une constante multiplicative près, est exactement (5). Alors que le second terme est une constante< 1 multipliée par $1/n$ (avec n grand).

Proposition 2.5.43 *Les matrices $S, J/n$ et G sont stochastiques, autrement dit, la somme des éléments d'une ligne quelconque vaut 1.*

Démonstration. Il s'agit d'une simple vérification. □

Lemme 2.5.44 *Si M est une matrice stochastique, alors 1 est valeur propre dominante de M.*

Démonstration. Soit $M \in \mathbb{Q}_r^r$. En multipliant tous les éléments de M par le p.p.c.m. γ des dénominateurs des éléments de M, la matrice

$$M' = \gamma M$$

obtenue est telle que la somme des éléments de chaque ligne vaut $\gamma \in \mathbb{N}$. Il s'agit donc de la matrice d'adjacence d'un graphe γ-régulier. Comme nous le verrons, il suffit d'appliquer la proposition II.4.3 pour voir que γM possède γ comme valeur propre dominante (i.e., toute autre valeur propre μ est telle que $|\mu| \leq \gamma$). La conclusion suit aisément en divisant par γ. Par construction, la matrice G est primitive puisque toutes ses entrées sont strictement positives. On peut appliquer le théorème de Perron, or par le lemme précédent, nous savons déjà que 1 est valeur propre dominante de G. Par conséquent, la valeur propre dominante 1 est simple et il existe un unique vecteur colonne $x > 0$ (resp. ligne $y > 0$) tel que

$$\sum_{i=1}^{n} x_i = 1 \left(\text{ resp. } \sum_{i=1}^{n} y_i = 1 \right) \quad \text{et} \quad Gx = x(\text{ resp. } yG = y).$$

Ainsi, déterminer le vecteur π des "PageRanks" revient à chercher le vecteur propre y de Perron à gauche de G (ou le vecteur propre de Perron à droite de $\widetilde{G}$). En appliquant le résultat asymptotique(3) énoncé à la page 84, puisque $e = (1 \cdots 1)^{\text{Cest un vecteur propre à droite de } G \text{ de valeur propre 1 } (G \text{ est stochastique})}$ et que $\pi.e = 1$ (puisque les scores sont normalisés), on a que

$$G^k = e\pi + o(1) \quad \text{autrement dit , } \lim_{k\to\infty} G^k = e\pi = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \begin{pmatrix} \pi_1 & \cdots & \pi_n \end{pmatrix}. \tag{2.6}$$

Nous pouvons à présent obtenir aisément une méthode itérative pour estimer π. Soit $p^{(0)} = \begin{pmatrix} p_1^{(0)} & \cdots & p_n^{(0)} \end{pmatrix} > 0$ un vecteur tel que $\sum_i p_i^{(0)} = 1$. Pour tout $k \geq 1$, on pose $p^{(k)} = p^{(0)}G^k = p^{(k-1)}G$. Il nous reste à montrer que

$$\lim_{k\to\infty} p^{(k)} = \pi$$

ainsi, il suffit dès lors de partir d'une distribution initiale et d'appliquer G de manière itérative jusqu'à obtenir la précision voulue mesurée par $p^{(k)} - p^{(k-1)}$ (différence en norme, bien entendu). C'est immédiat, au vu de (2.6),

13. La somme porte sur tous les puits et est donc indépendante de j.

$$\lim_{k\to\infty} G^k = \begin{pmatrix} \pi_1 & \pi_2 & \cdots & \pi_n \\ \vdots & \vdots & & \vdots \\ \pi_1 & \pi_2 & \cdots & \pi_n \end{pmatrix} =: P$$

et

$$\left[p^{(0)}P\right]_j = \sum_{i=1}^{n} p_i^{(0)}\pi_j = \pi_j \underbrace{\sum_{i=1}^{n} p_i^{(0)}}_{=1} = \pi_j.$$

Remarque 2.5.45 *En pratique, une centaine d'itérations suffisent pour obtenir une approximation utilisable et ce calcul peut être réalisé hors ligne, par exemple, une fois par mois, pour mettre à jour le vecteur des scores.*

Remarque 2.5.46 *En pratique, on se ramène à la matrice creuse H (possédant de nombreux zéros) :*

$$\begin{aligned} p^{(k)} &= p^{(k-1)}G \\ &= p^{(k-1)}\left(\alpha S + (1-\alpha)\frac{J}{n}\right) \\ &= \alpha p^{(k-1)}S + (1-\alpha)\frac{\widetilde{e}}{n} \\ &= \alpha p^{(k-1)}\left(H + a\frac{\widetilde{e}}{n}\right) + (1-\alpha)\frac{\widetilde{e}}{n} \\ &= \alpha p^{(k-1)}H + \left(\alpha p^{(k-1)}a + (1-\alpha)\right)\frac{\widetilde{e}}{n} \end{aligned}$$

où

$$a = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$$

est tel que $a_i = 1$ si $d^+(i) = 0$ et $a_i = 0$ si $d^+(i) > 0$.

2.6 Exercices récapitulatifs

Chapitre

3

Quelques problèmes et applications

3.1 Flot maximum

3.1.1 Généralités

La notion de flot dans un graphe est naturelle : étant donné un réseau de transport (train, tuyau, cables électriques), avec des capacités sur les arcs, quelle quantité de biens peut-on faire transiter au maximum ? Ou alors, en supposant qu'à chaque arc est attaché un coût unitaire, à quel coût minimum peut-on faire transiter un volume de biens donné ?
Une notion duale est celle de coupe dans un graphe. Une coupe est un ensemble d'arcs intersectant tout chemin entre deux sommets fixés. Les applications des coupes sont nombreuses en recherche opérationnelle bien sût, mais également en dehors de la recherche opérationnelle : positionnement de postes d'enquêtes, robustesse de réseaux, imagerie, etc.
La notion de flot ne peut exister que par rapport à un réseau. Avant de défnir ce qu'est un flot, il nous faut d'abord fixer un réseau. Soit donc un réseau modélisé par un graphe orienté $D = (V, A)$, muni d'une capacité $u \to \mathbb{R}_+$ et de deux sommets particuliers s (la source) et t (le puits). Une fonction $f : A \to \mathbb{R}_+$est un s-t flot si $f(a) \leq u(a)$ pour tout arc $a \in A$ et si pour tout sommet $v \notin \{s, t\}$, la loi de conservation

$$\sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a)$$

est respectée.
Etant donnée une $s - t$ coupe, il est intuitivement clair qu'un flot aura une valeur inférieure à sa capacité, où la capacité d'une s-t coupe $\delta^+(X)$ est définie par $\sum_{a \in \delta^+(X)} u(a)$. Le lemme suivant formalise entre autres cette propriété.

Lemme 3.1.1 *Soit $\delta^+(X)$ une s-t coupe et soit f un s-t flot quelconque. Alors*

$$\text{value}(f) = \sum_{a \in \delta^+(X)} f(a) - \sum_{a \in \delta^-(X)} f(a).$$

En particulier,

$$\text{value}(f) \leq \sum_{a \in \delta^+(X)} u(a). \tag{3.1}$$

La preuve de ce lemme est laissée en exercice.
On a également une propriété fondamentale qui est conforme à l'intuition qu'on peut se faire d'un flot : tout flot peut se décomposer en combinaison conique de s - t chemins élémentaires et circuits élémentaires.
Pour tout $B \subseteq A$, on note χ^B la fonction qui à $a \in A$ renvoie 1 si $a \in B$ et 0 sinon.

Proposition 3.1.2 *Soit f un s-t flot, $\mathcal{P}$ l'ensemble des s-t chemins élémentaires et $\mathcal{C}$ l'ensemble des circuits élémentaires. Il existe alors des coefficients réels positifs $(\lambda_P)_{P \in \mathcal{P}}$ et $(\mu_C)_{C \in \mathcal{C}}$ tels que*

$$f = \sum_{P \in \mathcal{P}} \lambda_P \chi^P + \sum_{C \in \mathcal{C}} \mu_C \chi^C$$

De plus,

$$\text{value}(f) = \sum_{P \in \mathcal{P}} \lambda_P.$$

La preuve n'est pas triviale, mais constitue tout de même un exercice raisonnable.

3.1.2 Flot maximum

Une question naturelleest celle de la valeur maximum d'un flot étant donné un réseau avec des capacités. Informellement, étant donné un réseau avec des capacités sur des arcs, la question est de savoir quelle quantité maximum de matière on peut faire passer de la source s à la destination t. Cette question se modélise de la façon suivante.

Problème du flot maximum

Donnée : Un graphe orienté $D = (V, A)$, deux sommets particuliers s (source) et t (puits) et des capacités $u : A \to \mathbb{R}_+$.
Tâche : Trouver un $s - t$ flot de valeur maximum.
En fait, ce problème est polynomial car il peut se modéliser comme un programme linéaire. La nature particulière du problème a poussé les chercheurs à concevoir des algorithmes ad hoc plus performants. L'un de ces algorithmes est celui d'Edmonds et Karp [7, 4], qui est également polynomial. Il est décrit dans le paragraphe suivant.
Un problème associé est celui de la $s - t$ coupe minimum. C'est en réalité le problème dual du problème du $s - t$ flot maximum, au sens de la dualité de la programmation linéaire.
Grosso-modo, le problème du flot maximal est un problème fondamental en théorie des graphes et en optimisation combinatoire. Il consiste à déterminer la quantité maximale de *flux* (ou de toute autre entité) pouvant être acheminée d'une source vers un puits dans un réseau de capacités, modélisé par un graphe orienté où chaque arc a une capacité limitée. Ce problème possède de nombreuses applications, allant de la planification de réseaux de transport et de communication à la gestion de flux de données.
La formulation du problème repose sur deux contraintes principales :

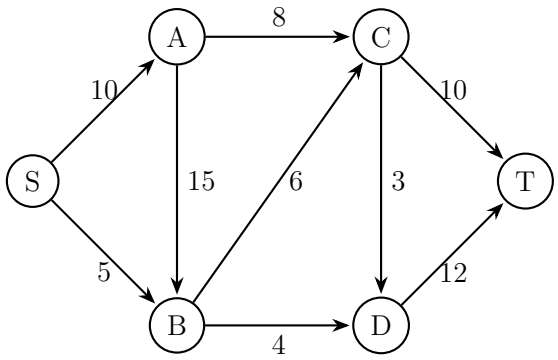
— les contraintes de capacité (le flot sur un arc ne peut excéder sa capacité) ;
— les contraintes de conservation (pour tout nœud (sommet) autre que la source et le puits, le flot entrant est égal au flot sortant).

Un des premiers algorithmes pour résoudre ce problème est l'algorithme **Ford-Fulkerson**, qui repose sur l'idée d'augmenter le flux existant en cherchant de nouveaux chemins augmentants entre la source et le puits. Tant qu'il existe un chemin dans le graphe résiduel (cest-à-dire le graphe indiquant la capacité restante sur chaque arête), on ajoute un flux égal à la capacité minimale le long de ce chemin. Ce processus se répète jusqu'à ce qu'aucun chemin augmentant n'existe plus. La complexité dépend de la méthode utilisée pour chercher ces chemins.

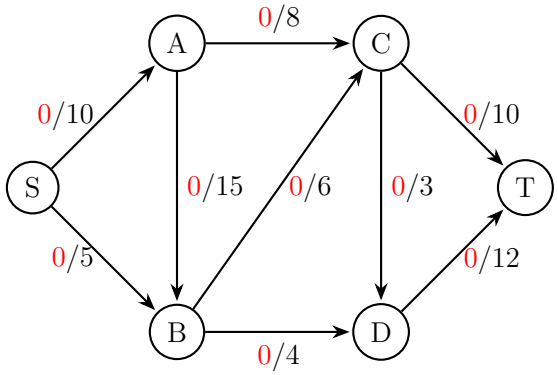
Algorithm 6 Ford-Fulkerson

1 : Initialiser $f(e) \leftarrow 0$ pour toute arête e
2 : **Tant que** il existe un chemin P de s à t dans le graphe résiduel **faire**
3 : $\quad c \leftarrow \min\{\text{capacité résiduelle}(e) : e \in P\}$
4 : $\quad$ **Pour** chaque arête e dans P **faire**
5 : $\quad\quad$ Augmenter $f(e)$ de c (ou diminuer si e est inversée)
6 : $\quad$ **Fin Pour**
7 : **Fin Tant que**
8 : **Retourner** f

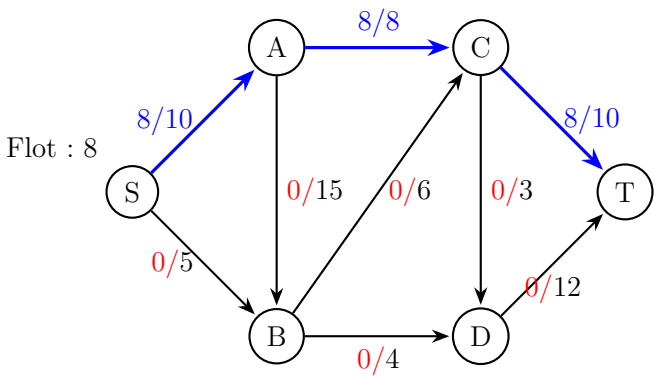
Exemple 3.1.3 *Considérons le réseau de flot suivant avec source S et puits T :*



Flot initial : 0 sur tous les arcs.



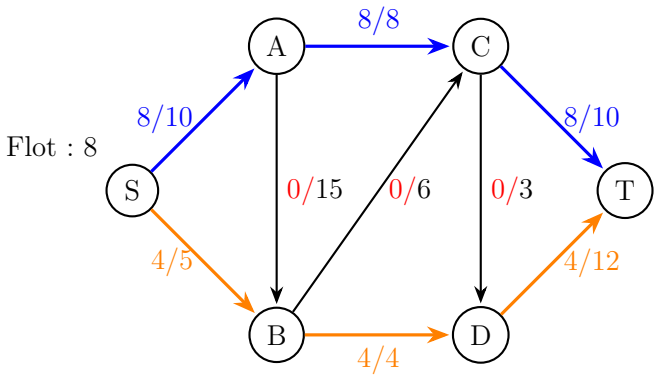
Itération 1 : Chemin $S \to A \to C \to T$ Capacité résiduelle minimale : $\min(10, 8, 10) = 8$



Après augmentation :

- $f(S, A) = 8$, capacité résiduelle : $10 - 8 = 2$
- $f(A, C) = 8$, capacité résiduelle : $8 - 8 = 0$
- $f(C, T) = 8$, capacité résiduelle : $10 - 8 = 2$

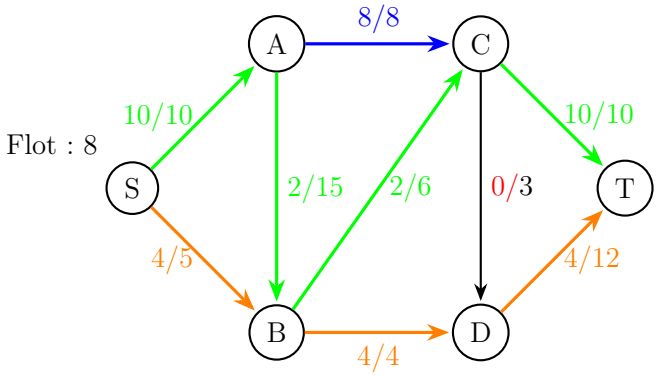
Itération 2 : Chemin $S \to B \to D \to T$ Capacité résiduelle minimale : $\min(5, 4, 12) = 4$.



Après augmentation :

- $f(S, B) = 4$, capacité résiduelle : $5 - 4 = 1$
- $f(B, D) = 4$, capacité résiduelle : $4 - 4 = 0$
- $f(D, T) = 4$, capacité résiduelle : $12 - 4 = 8$

Itération 3 : Chemin $S \to A \to B \to C \to T$ Capacité résiduelle minimale : $\min(2, 15, 6, 2) = 2$



Après augmentation :

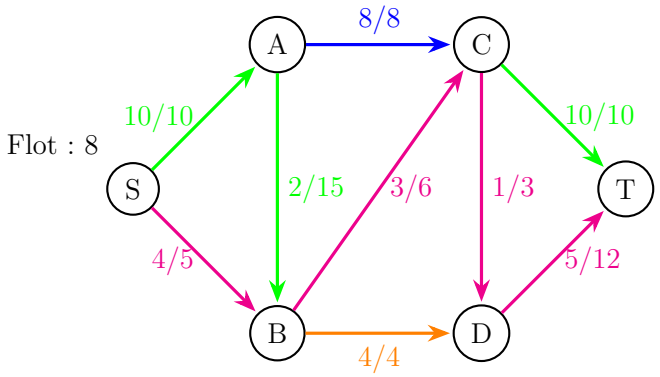
— $f(S,A) = 8+2 = 10$, capacité résiduelle : $2-2 = 0$

— $f(A,B) = 2$, capacité résiduelle : $15-2 = 13$

— $f(B,C) = 2$, capacité résiduelle : $6-2 = 4$

— $f(C,T) = 8+2 = 10$, capacité résiduelle : $2-2 = 0$

Itération 4 : Chemin $S \to B \to C \to D \to T$ Capacité résiduelle minimale : $\min(1,4,3,8) = 1$



Après augmentation :

— $f(S,B) = 4+1 = 5$, capacité résiduelle : $1-1 = 0$

— $f(B,C) = 2+1 = 3$, capacité résiduelle : $4-1 = 3$

— $f(C,D) = 1$, capacité résiduelle : $3-1 = 2$

— $f(D,T) = 4+1 = 5$, capacité résiduelle : $8-1 = 7$

Itération 5 : Aucun chemin augmentant Dans le graphe résiduel, il n'existe plus de chemin de S à T. L'algorithme se termine.
Le flot maximum est de 15.
Une amélioration importante est lalgorithme **Edmonds-Karp**, qui est une implémentation spécifique de Ford-Fulkerson utilisant une recherche en largeur (BFS) pour trouver les chemins augmentants les plus courts (en nombre darêtes). Cela garantit une borne de complexité polynomiale $O(VE^2)$, où V est le nombre de sommets et E le nombre darêtes. Ces méthodes reposent sur la notion de graphe résiduel et la mise à jour successive des flux, jusquà obtenir la valeur optimale du flot maximal.

Algorithm 7 Edmonds-Karp

```
1 : Initialiser f(e) ← 0 pour toute arête e
2 : Tant que il existe un chemin P de s à t trouvé par BFS dans le graphe résiduel faire
3 :     c ← min{capacité résiduelle(e) : e ∈ P}
4 :     Pour chaque arête e dans P faire
5 :         Augmenter f(e) de c (ou diminuer si e est inversée)
6 :     Fin Pour
7 : Fin Tant que
8 : Retourner f
```

3.2 Graphes bipartis : Problèmes de transport, mariages stables

3.2.1 Problème de transport

Le problème du transport est un programme linéaire qui a une structure particulière. Cette classe de PLs englobe les problèmes qui s'énoncent dans une forme approximative à celle-ci : ll y a m origines et n destinations, dans chaque origine on dispose d'une certaine quantité de matières premières (ou produit donné), et dans chaque destination on demande une certaine quantité de ce produit.
Le coût de transport est différent pour chaque couple origine-destination. On cherche un plan de transport optimal dans le sens qu'il minimise le coût total de transport. L'usage des tableaux de simplexe dans le cas des problèmes de transport est bien entendu possible. Toutefois, cette alternative ne présente pas un réel intérêt pratique car les problèmes de transport aboutissent généralement à un grand nombre de variables et de contraintes. Heureusement, une représentation intuitive et permettant un traitement facile des problèmes de transport existe : il s'agit du tableau de transport.

3.2.1.1 Représentation du problème de transport

Un problème de transport peut être représenté de trois manières :

— Sous la forme d'un tableau de transport

— Sous la forme d'un programme linéaire

— Sous la forme d'un graphe biparties

Formulation sous la forme d'un tableau de transport

Exemple 3.2.1 *Soit une série de villes alimentées en électricité par des centrales. La situation est résumée par la table suivante :*

	A	B	C	D	Puissance fournie (GWh)
1	6	5	3	1	500
2	10	8	4	2	300
3	7	9	11	12	200
Demande (GWh)	300	300	300	100	

TABLE 3.1 – Un tableau de transport.

La structure d'un tableau de transport est assez intuitive comme le montre l'exemple de le tableau 3.1.
Dans ce problème, on a trois origines et quatre destinations. Les offres des origines sont inscrites sur la dernière colonne, et les quantités disponibles dans les différentes destinations sont inscrites sur la dernière ligne. Les chiffres inscrits en petite taille dans chaque case indiquent les coûts de transport unitaires entre chaque origine et chaque destination. Par exemple, chaque unité transportée de l'origine 2 vers la destination 3 induit un coût de transport de 4(um). Remarquons que dans ce tableau l'offre totale est égale à la demande totale. On dit que ce problème est équilibré. Si le problème n'est pas équilibré, on peut toujours se ramener au cas équilibré en introduisant soit des origines ou des destinations fictives auquelles on attribue les surplus de l'offre (ou de la demande) avec des coûts nuls.

Formulation mathématique sous la forme d'un programme linéaire Définition des variables
x_{ij} = nombre de GWh produits à la centrale i et envoyé à la cité j
Description de la fonction économique

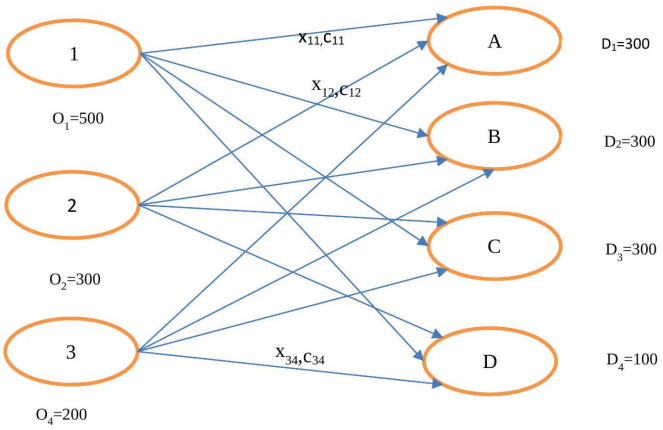
Min $Z = 6x_{11} + 5x_{12} + 3x_{13} + x_{14} + 10x_{21} + 8x_{22} + 4x_{23} + 2x_{24} + 7x_{31} + 9x_{32} + 11x_{33} + 12x_{34}$.

Les Contraintes de production
$$\begin{cases} x_{11} + x_{12} + x_{13} + x_{14} \leq 500 \\ x_{21} + x_{22} + x_{23} + x_{24} \leq 300 \\ x_{31} + x_{32} + x_{33} + x_{34} \leq 200 \end{cases}$$

Contraintes de consommation
$$\begin{cases} x_{11} + x_{21} + x_{31} \geq 300 \\ x_{12} + x_{22} + x_{32} \geq 300 \\ x_{13} + x_{23} + x_{33} \geq 300 \\ x_{14} + x_{24} + x_{34} \geq 100. \end{cases}$$

Plus les contraintes non négativité ($x_{ij} \geq 0$).

Formulation sous la forme de graphe bipartie



3.2.1.2 Résolution du Problème de transport

Comme dans la méthode du simplexe la résolution se déroule en deux parties :

1. Recherche d'une solution de base réalisable
2. Optimisation de la solution

Solution initiale Une solution de base pour un problème de transport avec m origines et n destinations doit contenir exactement : $m + n - 1$ variables de base.
Dans l'exemple proposé ci-haut on doit donc avoir dans tous les tableaux correspondants six variables de base.
Il y a plusieurs méthodes qui permettent d'obtenir une solution initiale. Nous présentons les deux méthodes les plus fréquemment utilisées.

La méthode du coin Nord-Ouest Partir du coin supérieur gauche du tableau.

1. Allouer le plus possible à la cellule courante et ajuster l'offre et la demande ;
2. Se déplacer d'une cellule vers la droite (demande nulle) ou le bas (offre nulle) ;

3. Répéter jusqu'au moment où toute l'offre est allouée et toute la demande est satisfaite.

	A	B	C	D	Offre
1	300 6	200 5	 3	 1	500
2	 10	100 8	200 4	2	300
3	 7	9	100 11	100 12	200
Demande	300	300	300	100	

TABLE 3.2 – Solution initiale

Le coût total correspondant à cette solution est de 6700(um). L'avantage principal de la méthode du coin nord-ouest est la facilité de mise en uvre. L'inconvénient majeur de cette méthode est qu'elle ne tient pas compte de la structure de coût. Généralement, mais pas systématiquement, elle aboutit à un coût total initial assez élevé.

La méthode du moindre coût L'idée consiste à exploiter les cases ayant des coûts de transport faibles et leur attribuer les quantités maximales (dans la mesure du possible). On procède de la manière suivante :

1. Repérer la case du tableau ayant le coût le plus faible ;
2. Affecter à cette case la quantité maximale possible ; une colonne ou une ligne est saturée :
3. Si une colonne est saturée, l'éliminer du tableau, mettre à jour la quantité dans la ligne correspondante et reprendre au point 1 avec le nouveau tableau ;
4. Si une ligne est saturée, l'éliminer du tableau, mettre à jour la quantité dans la colonne correspondante et reprendre au point 1 avec le nouveau tableau ;
   Lorsque toutes les lignes et toutes les colonnes sont saturées, le tableau doit contenir exactement ($\mathbf{m + n}$ - **1**) variables de base.

L'application à l'exemple proposé plus haut nous donne le tableau représenté dans le tableau 3.3.

	A	B	C	D	Offre
1	 6	100 5	300 3	100 1	500
2	100 10	200 8	4	2	300
3	200 7	9	11	12	200
Demande	300	300	300	100	

TABLE 3.3 – Solution intiale

Le coût total correspondant à cette solution initiale est de 5500(um).
En effet, de manière générale, la méthode du moindre coût aboutit à une meilleure solution initiale que la méthode du coin nord-ouest

Recherche de la solution optimale Afin de trouver le tableau optimal, on procède comme dans le simplexe classique. La première étape consiste à calculer les coûts marginaux pour les variables hors-base.
Il s'agit des δ_{ij} comme indiqué dans le tableau 3.4. Ce calcul se fait en trois étapes :

	A	**B**	**C**	**D**	Offre
1	δ_{11} **6**	**100** **5**	**300** **3**	**100** **1**	**500**
2	**100** **10**	**200** **8**	δ_{23} **4**	δ_{23} **2**	**300**
3	**200** **7**	δ_{32} **9**	δ_{33} **11**	δ_{34} **12**	**200**
Demande	**300**	**300**	**300**	**100**	

TABLE 3.4 – Les coûts marginaux

1. Pour chaque variable de base écrire l'équation : $c_{ij} = u_i + v_j$
2. Résoudre le système obtenu en fixant : $u_i = 0$
3. Calculer les valeurs des coûts marginaux à partir du système : $\delta_{ij} = C_{ij} - u_i - v_j$

$$
\begin{array}{lcll}
 & & u_1 = 0 & \\
u_1 + v_2 = 5 & \Rightarrow & v_2 = 5 & \\
u_1 + v_3 = 3 & \Rightarrow & v_3 = 3 & \delta_{23} = 4 - 3 - 3 = -2 \\
u_1 + v_4 = 1 & \Rightarrow & v_4 = 4 & \delta_{24} = 2 - 3 - 1 = -2 \\
u_2 + v_2 = 8 & \Rightarrow & u_2 = 3 & \delta_{32} = 9 - 0 - 5 = +4 \\
u_2 + v_1 = 10 & \Rightarrow & v_1 = 7 & \delta_{33} = 11 - 0 - 3 = +8 \\
v_1 + u_3 = 7 & \Rightarrow & u_3 = 0 & \delta_{34} = 12 - 0 - 1 = +11
\end{array}
$$

II est pratique d'effectuer ce calcul directement sur le tableau de transport.

	A	**B**	**C**	**D**	Offre	u_i
1	-1 **6**	**100** **5**	**300** **3**	**100** **1**	**500**	0
2	**100**	**200**	-2 **10**	-2 **8**	**300**	**3**
3	**200**	+4 **9**	+8 **11**	+11 **12**	**200**	0
Demande	**300**	**300**	**300**	**100**		
v_j	**7**	**5**	**3**	**1**		

Coût $= 100 * 5 + 300 * 3 + 100 * 1 + 100 * 10 + 200 * 8 + 200 * 7 = 5500$um Ce tableau contient des coûts marginaux strictement négatifs, et donc il ne s'agit pas d'une solution optimale.

— Si elle est optimale on s'arrête, sinon on refait une autre itération, et ainsi de suite.

— Ici, on peut prendre la variable x_{23} comme variable entrante.

— La valeur de X_{23} doit être augmentée mais tout en respectant les contraintes d'offre et de demande, ainsi que la non-négativité des autres variables.

	A	B	C	D	Offre	u_i
1	-1 6	100 + 5	300 - 3	100 1	500	0
2	100 10	200 - 8	-2 + 4	-2 2	300	3
3	200 7	+4 9	+8 11	+11 12	200	0
Demande	300	300	300	100		
v_j	7	5	3	1		

Sur le tableau 3.2.1.2, le cycle — indique les variations à effectuer pour sauvegarder une solution de base réalisable. On obtient ainsi le tableau donné par la figure 3.1., et pour lequel on refait le même travail, afin d'obtenir la solution optimale donnée par le tableau 3.2.

— Un signe " + " indique une augmentation de la quantité et un signe "-" indique une diminution de la quantité.
— En valeur absolue, la variation est la même pour toutes les cases sur le cycle.
— Dans notre cas :
— X_{23} augmente de 200 unités,
— X_{22} diminue de 200 unités,
— x_{12} augmente de 200 unités
— et enfin X_{13} diminue de 200 unités

	A	B	C	D	Offre	u_i
1	- 3 + 6	300 5	100 - 3	100 1	500	0
2	100 - 10	+2 8	200 + 4	0 2	300	1
3	200 7	+6 9	+10 11	+12 12	200	-2
Demande	300	300	300	100		
v_j	9	5	3	1		

FIGURE 3.1 – Le deuxième tableau de transport

Coût = 5200 um

	A	B	C	D	Offre	u_i
1	100 6	300 5	3	100 1	500	0
2	+3 10	+2 8	300 4	0 2	300	1
3	200 7	+3 9	+7 11	+10 12	200	1
Demande	300	300	300	100		
v_j	6	5	3	1		

FIGURE 3.2 – Le troisième tableau (optimal) Coût=4800 um

Casparticuliers Offre supérieure à la demande.
Afin de retrouver un problème de transport correctement structuré, on ajoute une demande fictive avec des coûts unitaires nuls, et dont la demande correspond à l'excédent de l'offre. Après cette transformation, un tableau de transport peut être construit et la résolution se fait de manière classique.
Offre inférieure à la demande.
Dans ce cas le problème n'admet pas de solution réalisable, et donc pas de solution optimale.

Dégénérescence. La Dégénérescence peut apparaître, soit au niveau de la recherche de la solution initiale, soit au cours des itérations pour la recherche de la solution optimale. Solution initiale : A chaque fois qu'on ajoute une quantité, soit une ligne soit une colonne est saturée. Il se peut toutefois que, simultanément, une ligne et une colonne soient saturées. Dans ce cas on obtiendra moins de variables de base. Afin de compléter la base, on ajoute une variable de base avec une quantité nulle.

3.2.2 Mariages stables

Ce paragraphe est entièrement tiré de [2].

3.2.2.1 L'agence matrimoniale honnête

L'agence matrimoniale que nous considérons est honnête dans le sens qu'elle prétend respecter les principes suivants :

1. l'information la plus large : tout adhérent a accès au fichier des adhérents de l'autre sexe, et peut les rencontrer ;
2. la stabilité des couples : l'agence ne forme que des couples stables (contrairement à d'autres agences, celle-ci ne vit pas de l'instabilité des couples) ;
3. l'efficacité : tout adhérent est assuré de rencontrer au moins un adhérent de l'autre sexe avec lequel il peut former un couple stable (contrairement à d'autres agences, personne ne cotise en pure perte).

Notre objectif dans cette première partie, sera de démontrer qu'une telle agence matrimoniale honnête n'existe pas, plus précisément que le "système d'axiomes" constitué par les trois principes énoncés

ci-dessus est contradictoire. A cet effet, nous allons d'abord leur donner un sens plus précis.

Position du problème. - Nous supposons que chaque adhérent classe ceux de l'autre sexe selon son ordre de préférences (il peut le faire d'après l'axiome d'information, et nous allons voir qu'il doit le faire pour satisfaire l'axiome de stabilité). Ainsi chaque garçon classe les filles selon un ordre total, et réciproquement. On obtient ainsi une configuration de préférences. En voici un exemple, avec n garçons et n filles (ici, $n = 3$) : On dit qu'on détermine une solution en mariant garçons et filles,

$G_1:$	F_1	F_2	F_3	$F_1:$	G_3	G_2	G_1
$G_2:$	F_1	F_3	F_2	$F_2:$	G_2	G_1	G_3
$G_3:$	F_2	F_3	F_1	$F_3:$	G_1	G_2	G_3

TABLE 3.5 – Configuration de préférences complètes P_1.

donc en formant n couples. Mais il existe deux catégories de solutions : les stables et les instables. Une solution instable est une solution telle qu'il existe un garçon G et une fille F non mariés ensemble qui se préfèrent mutuellement à leur épouse et époux respectifs. On dit que (G, F) est une cause d'infidélité ou une cause d'instabilité.

Exemple 3.2.2 *La solution* $(G_1, F_1), (G_2, F_3), (G_3, F_2)$ *est instable pour* P_1*, car* (G_2, F_1) *est une cause d'infidélité. En effet,* G_2 *préfère* F_1 *à son épouse* F_3*, et* F_1 *préfère* G_2 *à son époux* G_1

A l'inverse, une solution stable est une solution sans cause d'infidélité. Il suffit que chaque garçon constate que toute fille qu'il préfère à son épouse est mariée à un garçon qu'elle préfère à lui (il s'ensuit que la réciproque est vraie aussi).

Exemple 3.2.3 *La solution* $A = (G_1, F_2)(G_2, F_3)(G_3, F_1)$ *est stable pour la configuration* P_1*. En effet,* G_1 *préfère* F_1 *à son épouse mais ce n'est pas réciproque,* G_2 *est dans la même situation,* G_3 *aurait préféré* F_2 *ou* F_3 *mais ce n'est pas réciproque. Il est clair qu'on ne trouve aucune cause d'infidélité.*

Nous allons démontrer un peu plus loin qu'il existe toujours une solution stable, quelle que soit la configuration de préférences. Cet optimiste théorème d'existence semble a priori en contradiction avec la conclusion pessimiste annoncée plus haut. Il l'est en effet dans le cas très particulier que nous avons considéré ici, où d'une part il existe le même nombre d'adhérents pour chaque sexe, d'autre part chaque adhérent classe tous ceux de l'autre sexe, donc admet implicitement qu'il peut épouser chacun d'eux. Or ce "modèle idéal" ne reflète évidemment pas la réalité vécue.
Nous allons donc nous situer dans un cadre plus général, celui des configurations de préférences incomplètes. En voici un exemple :

Comme on le voit, chaque adhérent ne classe qu'une partie de ceux (ou celles) de l'autre sexe, seulement ceux qu'il juge acceptables. On peut aussi considérer qu'il a classé tous ceux de l'autre sexe, comme dans le cas des préférences complètes, mais qu'il s'est également classé lui-même parmi eux, en ce sens qu'il préfère les acceptables à lui-même (c'est-à-dire à son célibat), mais qu'il préfère son célibat plutôt qu'à former un couple avec un(e) inacceptable, selon le schéma suivant :

0. Le mot anglais est matching, son équivalent français habituel, le mot couplage, nous semble particulièrement malvenu dans le contexte et nous préférons donc parler de solution, celle-ci ne saurait être finale...

G_1 :	(F_5)	F_1	F_2	F_4		F_1 :	G_2	(G_4)	(G_1)	
G_2 :	F_2	(F_5)	F_1	F_3		F_2 :	G_1	G_2	G_3	
G_3 :	F_3	F_2	F_4	F_5	(F_1)	F_3 :	G_2	G_3	G_4	(G_1)
G_4 :	F_4	F_3	F_5			F_4 :	(G_2)	G_3	G_1	G_4
						F_5 :	G_3	G_4		

TABLE 3.6 – Configuration de préférences incomplètes P_2.

$$G : \underbrace{F \dots F}_{\text{acceptables}}\ G\ \underbrace{F \dots F}_{\text{inacceptables}}$$

Dans la configuration P_2 ci-dessus, nous n'avons reproduit que les listes d'acceptables, pour alléger et parce que seules ces listes importent.
Dans le contexte d'une configuration de préférences incomplètes pour m garçons et n filles, une solution est une partition de l'ensemble des adhérents en k couples (G, F) et $m + n - 2k$ célibataires (G) et/ou (F), où $k \leq \min(m, n)$ (non nécessairement égal). Ceux et celles qui restent célibataires s'appellent les laissés-pour-compte de la solution.
Une solution est dite irrationnelle si elle contient un couple dont l'un des membres est jugé inacceptable par l'autre. On rationnalise une configuration de préférences en mettant entre parenthèses sur la liste de chaque adhérent ceux de ses acceptables qui le jugent, lui, inacceptable. C'est ainsi que dans l'exemple de la configuration P_2, nous voyons que G doit mettre entre parenthèses les sentiments qu'il éprouve pour F_5.
La définition d'une solution instable est inchangée, à cela près qu'un célibataire peut entrer dans une cause d'infidélité.
Exemple : la solution $(G_1, F_1)\,(G_2, F_2)\,(G_3, F_5)\,(G_4, F_4)\,(F_3)$ est instable, car (G_3, F_3) est une cause d'infidélité. (Notons en passant qu'une solution irrationnelle n'est qu'un cas particulier de solution instable.)
Nous pouvons à présent énoncer et démontrer le théorème d'existence fondateur de la théorie.

3.2.2.2 Théorème d'existence.

Théorème 3.2.4 *(Gale-Shapley [5]). Quelle que soit la configuration de préférences P, il existe au moins une solution stable pour P. De manière equivalente, il existe toujours un couplage stable dans un graphe biparti. De plus, un tel couplage se trouve en $O(nm)$.*

Démonstration. La démonstration repose sur un algorithme de construction d'une solution stable que nous appellerons algorithme des initiatives masculines. Nous supposons que l'agence organise un bal au cours duquel tous les adhérents se retrouvent pour danser. Les garçons invitent les filles au cours de mouvements successifs.
Au premier mouvement, chaque garçon invite la fille qu'il préfère. Si une fille se trouve alors prise entre plusieurs feux, elle choisit pour danser celui qu'elle préfère parmi ceux qu'elle juge acceptables, et qui devient par le fait même son prétendant, tandis que les autres sont rejetés par la fille. Ces garçons, non prétendants à l'issue du premier mouvement, sont dits libres.
Au deuxième mouvement, chaque garçon libre invite la fille qui suit dans l'ordre de ses préférences, même si elle possède déjà un prétendant. Chaque fille choisit à nouveau celui qu'elle préfère entre ceux qui l'invitent et son éventuel prétendant, et désigne ainsi un nouveau prétendant, les autres étant rendus à leur liberté.

Et ainsi de suite. . . L'algorithme s'arrête lorsqu'il n'y a plus de garçons libres (ce qui se produit au bout d'un nombre fini de mouvements, puisqu'il n'y a qu'un nombre fini de garçons, dont les listes de filles acceptables sont de longueurs finies). On marie alors chaque fille avec son prétendant. Reste célibataire tout garçon qui s'est fait rejeter par toutes les filles qu'il juge acceptables, reste célibataire aussi toute fille qui n'a été invitée par aucun des garçons qu'elle juge acceptables. Nous notons Masc la solution ainsi définie.
Montrons que Masc est une solution stable. Soit G un garçon quelconque, et F une fille qu'il préfère à son épouse (ou à son célibat, dans le cas où Masc le désigne comme célibataire). Il est clair que F l'a rejeté au cours de l'algorithme, puisqu'il a invité toutes les filles dans l'ordre décroissant de ses préférences. Elle l'a rejeté au profit d'un certain G' qu'elle préfère à G, or son époux à l'issue de l'algorithme est soit G', soit un autre qu'elle aime encore plus, car les prétendants qui se sont succédés auprès d'elle allaient dans l'ordre croissant de ses préférences. Il est donc exclu qu'elle préfère G à son époux, autrement dit (G, F) n'est pas une cause d'infidélité. Par conséquent, la solution Masc est stable. □

Sous forme d'algorithme, le théorème précédent est repris dans le pseudocode suivant :
Voyons par exemple ce que donne l'algorithme des initiatives masculines sur la configuration P_1 :

Algorithm 8 Algorithme de Gale-Shapley

Exiger Deux ensembles H (hommes) et F (femmes), préférences de chacun
S'assurer que Affectation stable
1 : Initialiser tous les hommes et femmes comme célibataires
2 : Créer une file contenant tous les hommes célibataires
3 : **Tant que** la file n'est pas vide **faire**
4 : $h \leftarrow$ premier homme de la file
5 : $f \leftarrow$ femme préférée de h qu'il n'a pas encore demandée
6 : **Si** f est célibataire **alors**
7 : h et f deviennent engagés
8 : Retirer h de la file
9 : **Sinon**
10 : $h' \leftarrow$ fiancé actuel de f
11 : **Si** f préfère h à h' **alors**
12 : Rompre l'engagement de f avec h'
13 : h et f deviennent engagés
14 : Retirer h de la file
15 : Ajouter h' à la file
16 : **Sinon**
17 : h reste dans la file
18 : **Fin Si**
19 : **Fin Si**
20 : **Fin Tant que**
21 : **Retourner** les couples formés

au premier mouvement, G_1 et G_2 invitent F_1, qui choisit G_2 et rejette G_1, tandis que G_3 invite F_2. Au deuxième mouvement, le garçon libre G_1 invite F_2, laquelle le préfère au détriment de G_3. Au

troisième mouvement, celui-ci se console avec F_3. D'où le résultat, en tête du tableau suivant :

$$\begin{aligned}\text{Masc}\,(P_1) &= (G_1, F_2)\,, (G_2, F_1)\,, (G_3, F_3)\\ A &= (G_1, F_2)\,, (G_2, F_3)\,, (G_3, F_1)\\ \text{Fem}\,(P_1) &= (G_1, F_3)\,, (G_2, F_2)\,, (G_3, F_1)\end{aligned}$$

En dessous de Masc, nous avons fait figurer la solution A proposée en début d'article comme exemple de solution stable pour P_1, puis la solution Fem qui résulte de manière analogue (en un seul mouvement) d'un algorithme des initiatives féminines où les rôles sont renversés. Ainsi nous avons sur cet exemple, parmi les $3! = 6$ solutions possibles en trois couples (les solutions avec célibataires sont évidemment instables, puisque les préférences sont complètes), trois solutions stables, et il est facile de vérifier que ce sont les seules.
Quelle solution stable choisir lorsque plusieurs se présentent à nous ? Sur cet exemple nous pouvons constater que des conflits entre les sexes apparaissent dès qu'on aborde cette question, en l'occurrence les garçons préfèrent Masc, les filles Fem (alors que pour les garçons c'est la pire des solutions envisageables), et A occupe une position intermédiaire. Nous allons voir que cette situation de désaccord est générale, sauf si Masc = Fem, ce qui peut se produire.

Prenons le deuxième exemple suivant :

Hommes : A, B, C, Femmes : X, Y, Z
Préférences :

— A : Y > X > Z
— B : X > Y > Z
— C : X > Y > Z
— X : B > A > C
— Y : A > C > B
— Z : A > B > C

Déroulement :

1. A demande à Y engagé (A-Y)
2. B demande à X engagé (B-X)
3. C demande à X X préfère B C rejeté
4. C demande à Y Y préfère A C rejeté
5. C demande à Z engagé (C-Z)

Solution stable : (A-Y), (B-X), (C-Z).

Nous montrerons un peu plus loin que les garçons préfèrent Masc (et les filles Fem) à toute autre solution stable, selon le théorème suivant :

Théorème 3.2.5 *d'optimalité (Gale-Shapley). - Soit une configuration de préférences P et soit* Masc(P) *la solution correspondante. Alors tout garcon est marié dans* Masc(P) *à la fille qu'il préfère parmi toutes celles qu'il épouserait selon les diverses solutions stables pour la configuration P.*

Désaccords conjugaux. Laissés-pour-compte. - Nous allons à présent aborder deux théorèmes pessimistes. Commençons par établir que si A et B sont deux solutions stables pour une configuration de préférences P (complète ou incomplète), alors deux époux ne sont jamais d'accord sur la préférence à accorder à l'une ou l'autre de ces deux solutions, sauf lorsque cela leur est indifférent à tous deux parce qu'ils sont mariés de la même manière dans les deux solutions. Lorsque nous dirons qu'un conjoint préfère A à B, nous sous-entendrons strictement (il n'est pas marié de la même manière selon les deux solutions et préfère son conjoint dans A à son conjoint dans B).

Théorème 3.2.6 *des désaccords conjugaux. - Soient A et B deux solutions stables pour une configuration de préférences P, et soit G un garçon qui préfère A. Alors il est marié dans A et son épouse F dans A préfère B. En outre il est aussi marié dans B et son épouse F' dans B préfère également B.*

Démonstration. Il est évident que G est marié dans A. Si G et F préféraient tous deux A, ils seraient une cause d'infidélité pour B, où ils ne sont pas mariés, mais B est stable. Donc F préfère B. Désignons par $\Gamma(A)$ l'ensemble des garçons qui préfèrent A et par $\Phi(B)$ l'ensemble des filles qui préfèrent B. Il résulte de ce qui précède que le mariage dans A est une application injective de $\Gamma(A)$ dans $\Phi(B)$. Pour une raison symétrique, le mariage dans B est une application injective de $\Phi(B)$ dans $\Gamma(A)$ (sinon, c'est A qui serait instable). Par conséquent les deux ensembles $\Gamma(A)$ et $\Phi(B)$ ont même cardinal et les deux applications sont en fait des bijections. Par suite G est bien marié dans B, et avec une fille de $\Phi(B)$, ce qui achève la démonstration du théorème. □

Théorème 3.2.7 *des laissés-pour-compte (Gale-Sotomayor). - Soit P une configuration de préférences incomplètes, l'ensemble des laissés-pourcompte (c'est-à-dire des non marié(e)s dans une solution) ne dépend pas de la solution stable choisie.*

Démonstration. En effet, supposons que G soit laissé-pour-compte dans la solution stable B, et soit A une autre solution stable quelconque. Si G était marié dans A il préférerait A à B, mais nous venons de voir qu'il serait également marié dans B ce qui est contradictoire. Donc G est exclu de toute solution stable. Le même raisonnement s'applique symétriquement à une fille F. □

Exemple 3.2.8 *dans le cas de la configuration incomplète P_2, les quatre solutions stables (dont la liste sera donnée au paragraphe suivant) désignent la même laissée-pour-compte, F_5.*

Le théorème des laissés-pour-compte montre bien que le troisième axiome de l'agence matrimoniale honnête est en contradiction avec les deux autres. Deux remèdes sont envisagés en pratique : soit on fait une entorse à l'axiome d'information, en privilégiant dans les rencontres certains clients au détriment d'autres, soit on fait patienter en attendant de nouveaux adhérents de l'autre sexe.

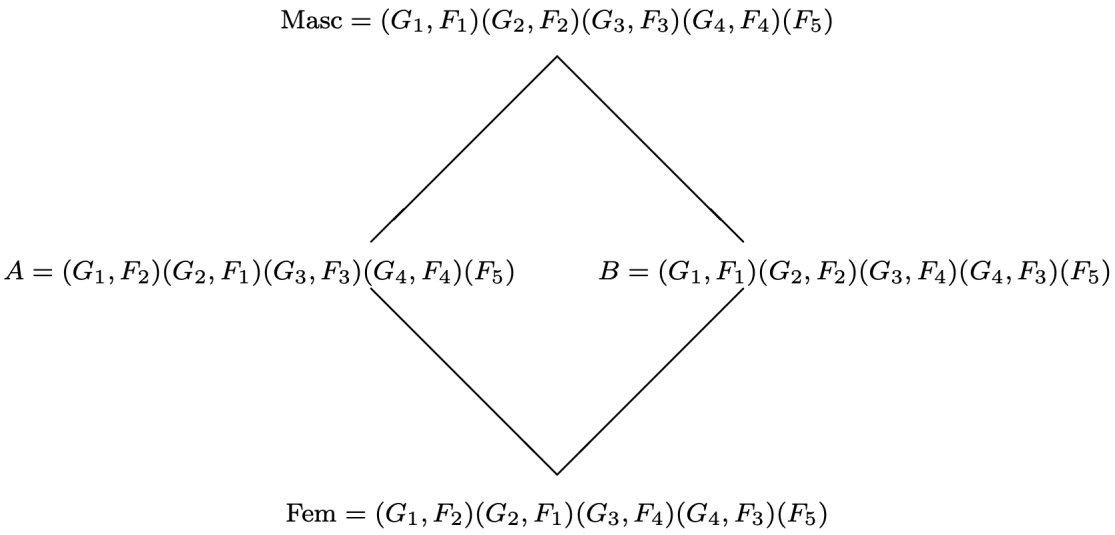
Treillis des solutions stables. - Une configuration de préférences complètes ou incomplètes P étant donnée, il existe un certain ensemble de solutions stables. Nous avons vu que chaque adhérent (garçon ou fille) munit cet ensemble d'une relation d'ordre partielle, celle de ses préférences entre les solutions. Considérons la relation d'ordre partielle de préférence masculine, où l'on dit que A est préférée par les garçons à B si et seulement si chacun d'eux soit préfère A à B, soit est indifférent (autrement dit $\Gamma(B)$ est vide). Bien entendu d'après le théorème des désaccords conjugaux B est préférée par les filles à A, car $\Phi(A)$ est vide. Cette relation d'ordre est partielle car il peut arriver que $\Gamma(A)$ et $\Gamma(B)$ soient simultanément non vides. C'est le cas plus loin dans l'exemple de P_2, contrairement au cas de P_1, où l'ordre entre les trois solutions stables était total.

Théorème 3.2.9 *(Conway). - La relation d'ordre de préférence masculine définit une structure de treillis sur l'ensemble des solutions stables pour une configuration P donnée.*

Théorème 3.2.10 *En effet, soient A et B deux solutions stables quelconques. Désignons par C la solution qui consiste à attribuer à chaque garçon celle qu'il préfère entre ses deux épouses selon A et selon B.*

3.2.2.3 Mariages stables

Montrons d'abord qu'on définit bien ainsi une solution, c'est-à-dire que deux garçons G et G' ne risquent pas de se voir attribuer la même fille F. Si cela était, celle-ci serait par exemple l'épouse de G dans A et de G' dans B. Alors G préférerait A, et G' préférerait B, elle devrait donc être en désaccord avec tous les deux sur la préférence à accorder entre les deux solutions, ce qui est absurde. Donc C est bien une solution. Notons en passant que C consiste à faire épouser à chaque fille le garçon qu'elle aime le moins entre ses époux selon A et B.
Montrons que C est stable. Soit G un garçon quelconque, F une fille qu'il préfère à son épouse dans C, autrement dit à ses deux épouses dans A et dans B. Mais nous savons que F préfère son époux dans A à G, car A est stable, et elle préfère également son époux dans B à G, car B est stable. Or son époux dans C est l'un d'eux, celui des deux qu'elle aime le moins mais qu'elle préfère quand même à G. Donc C est stable. On désigne cette solution C par $\sup(A, B)$ (sup pour les garçons, bien sûr), on définit de la même manière une solution $\inf(A, B)$, et on vérifie sans difficulté que l'ensemble des solutions stables muni de ce sup et de cet inf est un treillis distributif.
Exemple : voici le treillis des quatre solutions stables de P_2, où Masc $= \sup(A, B)$ et Fem $= \inf(A, B)$:



Pour en revenir au cas général, le treillis étant fini, il existe toujours une solution maximum Max, et on peut reformuler le théorème d'optimalité annoncé plus haut sous la forme suivante :
Max = Masc.
Nous démontrerons ce théorème comme corollaire d'un résultat plus précis dû à Hwang, résultat qui sera l'outil central pour comprendre l'inefficacité du mensonge masculin.

Infidélité de l'épouse d'un revendicateur. - Soit P une configuration de préférences (complète ou non), Masc la solution stable issue de l'algorithme des initiatives masculines, et A une autre solution rationnelle. Un garçon R est dit revendicateur de A s'il préfère cette solution à Masc, parce qu'il est marié dans A à une fille qu'il préfère à son épouse selon Masc. On désigne par $R(A)$ l'ensemble des revendicateurs de A.

Une autre formulation du théorème d'optimalité consiste à dire que si $R(A)$ est non vide, alors A est instable. On peut cependant donner un résultat plus précis.

Théorème 3.2.11 *(Hwang [2].) - Soit A une solution rationnelle distincte de Masc. Si l'ensemble $R(A)$ des revendicateurs de A est non vide, alors il existe également des non revendicateurs et l'un d'eux, G, constitue une cause d'infidélité pour A avec l'épouse F d'un revendicateur.*

Démonstration. Notons d'abord que l'hypothèse que A est rationnelle implique que les revendicateurs sont jugés acceptables par leurs épouses dans A, et cela implique qu'elles sont également mariées dans Masc sinon Masc serait instable.
Il y a un cas où le théorème est évident, c'est quand il existe une fille F qui est l'épouse d'un revendicateur R dans A et l'épouse d'un non revendicateur G dans Masc. En effet il est clair que (G, F) est une cause d'infidélité : d'une part, G ne revendique pas A, et préfère donc F à son épouse (ou à son célibat) selon A, d'autre part F préfère G à R, sinon (R, F) serait une cause d'infidélité pour Masc, qui est stable.
Supposons à présent qu'une telle fille n'existe pas. Il en résulte que l'ensemble des épouses des revendicateurs de A est le même ensemble dans les deux solutions, et que ceux-ci revendiquent "simplement" de pouvoir échanger leurs épouses entre eux !
Reportons-nous au déroulement de l'algorithme des initiatives masculines et considérons le dernier mouvement au cours duquel on trouve un revendicateur invitant une fille. Soit R ce revendicateur et F cette fille. Il est clair que R épouse F à l'issue de Masc, sinon ce ne serait pas le dernier mouvement de ce type puisque nous savons que de toute manière R se marie à l'issue de Masc. Donc R n'est pas le revendicateur qui revendique F comme épouse dans A, sinon il serait indifférent et non revendicateur. Celui qui la revendique dans A est un certain R', qui épouse une autre fille F' à l'issue de Masc, mais qui préfère F à cette F' et qui par conséquent s'est fait rejeter par F au cours de l'algorithme. De cela on déduit que
F avait un prétendant à l'arrivée de R. Appelons G ce prétendant qui s'est fait évincer par R et montrons que (G, F) est la cause d'infidélité annoncée. En fait, le prétendant G n'est pas R', ni aucun autre revendicateur, sinon ce revendicateur reviendrait inviter ultérieurement dans le cours de l'algorithme, ce qui est exclu puique c'est au cours du dernier mouvement d'invitation par un revendicateur que G s'est fait supplanter. G est par conséquent un non revendicateur (il en existe donc). Puisque G est l'avantdernier prétendant de F, elle le préfère à R', qu'elle a rejeté antérieurement et qui veut être son époux dans A. D'autre part G préfère F à son épouse dans Masc (ou à son célibat) puisque F l'a rejeté. Enfin, G aime au moins autant sa situation dans Masc (épouse ou célibat) à sa situation dans A puisqu'il est non revendicateur. Donc (G, F) est une cause d'infidélité pour A. □

En même temps nous avons donc démontré le théorème d'optimalité, pour lequel il existe des démonstrations plus directes [6] mais aucune vraiment facile. Cependant le principal intérêt du théorème de Hwang est qu'il permet d'obtenir immédiatement le théorème de Dubins et Freedman dont la preuve originale était longue et technique.

3.2.2.4 Mensonges et manipulations

Prévoir le destin de chacun . - Nous supposons à présent que l'agence matrimoniale est dirigée par une dame plutôt conformiste qui considère comme allant de soi qu'on doit laisser toutes les initiatives aux garçons, et que par conséquent dès que la configuration de préférences P est connue, on détermine la solution stable qui lui correspond et qui n'est autre que $\text{Masc}(P)$, laquelle se dégagera tout naturellement au cours du bal qu'elle compte organiser, au cours duquel les "convenances" exigent que ce soient les garcons qui invitent.
Comme cette dame est curieuse de nature, elle exige de tous ses adhérents qu'ils expédient préalablement leurs listes de préférences (qu'ils auront établies après consultations du fichier et des rencontres au siège de l'agence, car nous supposons que la directrice respecte au moins l'axiome d'information). Ainsi, prenant connaissance de la configuration de préférences, elle pourra prévoir le déroulement de l'algorithme et connaitra avant tout le monde le destin de chacun.
Le prix de cette curiosité, c'est évidemment le risque d'indiscrétion. On peut imaginer que la secrétaire de l'agence communique tout ou partie de la configuration des préférences à l'un(e) des adhérent(e)s, permettant ainsi que des manipulations faussent le résultat.

Manipulations féminines. - Supposons par exemple que nous ayons trois garçons et trois filles et la configuration de préférences complètes que voici :

$G_1:$	F_1	F_2	F_3	$F_1:$	F_2	G_1	G_3
$G_2:$	F_2	F_1	F_3	$F_2:$	G_1	G_2	G_3
$G_3:$	F_2	F_3	F_1	$F_3:$	G_3	G_1	G_2

Configuration des vraies préférences P_3
Supposons que F_2 tarde à expédier sa liste, et se fasse discrètement communiquer par la secrétaire les listes de préférences des autres adhérents. Elle est en mesure de prévoir le déroulement de l'algorithme des initiatives masculines : au premier mouvement, G_1 devient prétendant de F_1, tandis que G_2 et G_3 sont en concurrence auprès de F_2 qui choisit G_2 ; au deuxième mouvement, G_3 revient auprès de F_3. D 'où :

$$\text{Masc}\,(P_3) = (G_1, F_1)\,(G_2, F_2)\,(G_3, F_3)\,.$$

C'est pourquoi F_2 décide de mentir sur ses préférences, et d'agir comme si elle préférait G_3 à G_2. D'où la configuration des fausses préférences que voici :

$G_1:$	F_1	F_2	F_3	$F_1:$	G_2	G_1	G_3
$G_2:$	F_2	F_1	F_3	$F_2:$	G_1	G_3	G_2
$G_3:$	F_2	F_3	F_1	$F_3:$	G_3	G_1	G_2

Configuration des fausses préférences P_4
Voici en effet le déroulement de l'algorithme sur la configuration P_4 : au premier mouvement, G_1 devient prétendant de F_1, tandis que G_2 et G_3 sont en concurrence auprès de F_2 qui choisit cette fois G_3 en mentant sur sa préférence. Au deuxième mouvement, G_2 invite F_1, qui le préfère et rejette donc G_1. Au troisième mouvement, G_1 vient auprès de F_2 qui triomphe car elle obtient enfin celui qu'elle préfère. Elle rejette G_3 qui se consolera au quatrième mouvement avec F_3. Résultat :

$$\text{Masc}\,(P_4) = (G_1, F_2)\,(G_2, F_1)\,(G_3, F_3)\,.$$

Autrement dit F_2 obtient par son mensonge la solution optimale pour les filles, qui n'est autre que Fem (P_3) et qui est donc stable, même pour les vraies préférences.

Certes, on peut voir que ce type de manipulation féminine n'est pas toujours possible avec une configuration complète. Cependant, dans le cas où l'agence autorise les configurations de préférences incomplètes (ce qui est bien la moindre des choses, le mariage forcé n'étant guère en vogue de nos jours), il est facile de voir que toute configuration est manipulable par les filles. Si les filles connaissent les préférences des garcons (et font patienter pour faire connaître les leurs), elles peuvent se réunir, se dévoiler clandestinement leurs vraies préférences, étudier la configuration des vraies préférences P, déterminer Fem (P) par l'algorithme des initiatives féminines. Puis elles décrètent inacceptable tout garçon qui vient après celui qui leur est attribué selon Fem(P). Il est clair que sur la configuration incomplète P' ainsi modifiée, qu'elles présenteront à l'agence, on a, inévitablement, Masc (P') = Fem (P') = Fem(P).

Impuissance du mensonge masculin. - On pourrait penser qu'un garçon, ou une coalition de garcons, peut en faire autant, et obtenir une fille préférable en mentant sur ses (ou leurs) préférences. Or nous pouvons déjà prévoir une première faiblesse du mensonge masculin, qui est que de toute manière en prétendant "améliorer" encore la solution Masc, qui est la solution optimale pour eux, il ne peut conduire qu'à des solutions instables pour les vraies préférences, or ce défaut majeur n'affectait nullement le mensonge féminin comme nous venons de le voir.
On peut cependant imaginer le cas d'un revendicateur indifférent au fait que la solution altérée par son mensonge soit instable, d'autant que rien ne lui prouve que c'est son épouse qui sera infidèle (le théorème de Hwang n'exclut pas que ce soit l'épouse d'un autre revendicateur). Les tentatives manipulatoires de ce genre d'individu seront ruinées par le théorème suivant.

Théorème 3.2.12 *(Dubins-Freedman [3]) - Si un garçon ment sur ses préférences au cours de l'algorithme des initiatives masculines, il n'obtiendra pas une fille préférable à celle qu'il aurait obtenu en agissant selon ses vraies préférences.*

Démonstration. Soit P la configuration des vraies préférences, P' une configuration modifiée qui ne diffère de P que par la liste des préférences d'un garçon R, que nous désignons ainsi parce que nous supposons qu'il préfère Masc (P') à Masc(P).

D'après le théorème de Hwang, A = Masc (P') est instable pour P, la cause d'instabilité étant un couple (G, F), où G est distinct de R. Or les préférences de G et de F selon P' sont rigoureusement les mêmes que selon P, donc la solution A est également instable pour P', ce qui est absurde puisque c'est Masc (P'). □
Ce théorème illustre en quel sens on peut dire qu'on trouve d'un côté ceux qui prétendent décider, et de l'autre, celles qui peuvent manipuler.

3.2.2.5 Problèmes de recherche

On peut compliquer ces problèmes de manipulation, liés à la théorie des jeux, par exemple en introduisant des dots, dans un sens ou dans l'autre, qui font varier la configuration de préférences. Citons à ce sujet les travaux de Gabrielle Demange [2, 1]. Sur le strict plan combinatoire et algorithmique, beaucoup d'aspects du sujet restent méconnus. On trouvera une liste de problèmes de recherche sur les mariages stables dans le livre de Knuth. En voici quelques-uns :

(1) Existe-t-il des "algorithmes d'initiatives mixtes" permettant d'engendrer les solutions stables intermédiaires, ou n'a-t-on pas d'autre ressource pour en dresser la liste que de tester toutes les solutions, en cherchant systématiquement pour chacune d'elle une éventuelle cause d'infidélité ?
(2) Avec n garçons et n filles, comment fabriquer une configuration de préférences complètes qui maximise le nombre de solutions stables ? Ce nombre maximal est 3 pour $n = 3$, très probablement 10 pour $n = 4$. Quelle est la croissance de ce nombre maximal en fonction de n ? On sait montrer qu'elle est au moins exponentielle, peut-on espérer qu'elle soit de l'ordre de $n!$ (nombre total de solutions) ?

Etat actuel des ces deux questions Les problèmes évoqués par Knuth concernant les mariages stables restent, pour la plupart, partiellement ouverts à ce jour. Bien que des progrès significatifs aient été réalisés depuis la publication de son ouvrage, plusieurs questions combinatoires et algorithmiques fondamentales n'ont pas été entièrement résolues.

Problème (1) : Algorithmes d'initiatives mixtes Ce problème reste substantiellement ouvert. Bien que des algorithmes efficaces aient été développés pour énumérer *toutes* les solutions stables (tels que l'algorithme d'énumération d'Irving et Leather), la question de savoir s'il existe des méthodes plus intelligentes, ń d'initiative mixte ż, pour générer des solutions stables intermédiaires sans avoir recours à une énumération exhaustive n'est pas complètement trancher. Des travaux récents en théorie des jeux et en optimisation discrète ont abordé ce problème sous l'angle de la ń génération implicite ż, mais une procédure générale et efficace fait toujours défaut.

Problème (2) : Nombre maximal de mariages stables Des progrès notables ont été accomplis sur ce front, mais la question centrale demeure ouverte.

- Pour $n = 4$, le nombre maximal est maintenant **connu** : il est de **10**, comme Knuth le soupçonnait [6, 1].
- Pour $n = 5$, la valeur exacte a été un sujet de recherche active. Le nombre maximal est maintenant établi à **26**.
- Pour $n = 6$, la valeur exacte est **inconnue**. La meilleure borne inférieure connue est de 91, mais la valeur maximale possible pourrait être plus élevée.

Croissance en fonction de n La croissance du nombre maximal de mariages stables en fonction de n est un problème majeur et toujours non résolu.

- Il a été prouvé que cette croissance est **au moins exponentielle**. Une construction explicite due à Knuth lui-même donne une borne inférieure de l'ordre de 2.28^n.
- La question de savoir si elle peut être de l'ordre de $n!$ (factorielle) a été résolue par la négative. Il a été démontré que le nombre de mariages stables pour toute instance est **au plus** $2^n - 1$, ce qui est une borne bien en deçà de $n!$.
- La **meilleure borne supérieure générale connue** est effectivement $O(2^n)$. La quête de la borne supérieure exacte et de constructions atteignant une croissance exponentielle aussi élevée que possible reste un domaine de recherche actif.

Autres questions ouvertes contemporaines Au-delà des problèmes historiques de Knuth, la recherche sur les mariages stables se poursuit sur de nombreux fronts :

— **Complexité moyenne** : Comprendre le comportement typique (et non plus pire cas) du nombre de mariages stables pour des préférences aléatoires.

— **Extensions au cas incomplet** : Étudier le nombre et la structure des mariages stables lorsque les listes de préférences peuvent être incomplètes.

— **Jeux de préférences** : Comment un agent peut-il modifier stratégiquement ses préférences pour obtenir un meilleur partenaire dans l'algorithme de Gale-Shapley ? Ce problème algorithmique et stratégique est profond.

— **Généralisations** : Ces questions sont étendues à des modèles plus généraux comme le problème des mariages stables avec appariements multiples (ń many-to-many ż) ou dans des contextes avec contraintes supplémentaires.

3.3 Problème de sac-à-dos et du bin-packing

Le thème de cette section est le suivant : on a des objets et des conteneurs, comment remplir au mieux ? Ecrit comme cela, le problème est assez imprécis. Nous allons nous focaliser sur deux problèmes particuliers qui rentrent dans la catégorie des problèmes de remplissage, le problème du sac-à-dos et celui du bin-packing. Le problème du sac-à-dos a déjà été vu au Chapitre 3, mais allons discuter d'autres aspects de ce problème.
Le problème du sac-à-dos dans sa version la plus simple peut se décrire informellement de la manière suivante : on a des objets de poids et de valeur variable ; on dispose d'un seul conteneur (le sac-à-dos) qui est muni d'une contrainte de poids ; remplir le conteneur de manière à maximiser la valeur des objets stockés.
Le problème du bin-packing dans sa version la plus simple peut se décrire informellement de la manière suivante : on a des objets de taille variée et un seul type de conteneur ; trouver le nombre minimum de conteneur permettant de tout stocker.
Ces problèmes ont des applications directes dans le domaine de la logistique : stocker des produits, remplir des camions, etc.

3.3.1 Problème du sac-à-dos

De façon formelle, le problème du sac-à-dos s'écrit :
Donnée : des entiers $n, p_1, \ldots, p_n$ et P, et des réels $c_1, \ldots, c_n$.
Tâche : trouver un sous-ensemble $S \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in S} p_j \leq P$ et $\sum_{j \in S} c_j$ est maximum.
Cela peut s'interpréter de la manière suivante : P est la charge maximale du conteneur, p_i est le poids de l'objet i, et c_i sa valeur. On a n objets, mettre dans le conteneur un sous-ensemble S d'objets de valeur maximale, tout en respectant la contrainte de poids.
Les applications en logistique sont évidentes. Mais il existe bien d'autres domaines où ce problème se retrouve. Par exemple, en finance : on a un budget fini P, on a des produits financiers i coûtant chacun p_i et rapportant c_i sur l'année à venir ; maximiser le profit.
Commençons par cerner la complexité du problème.

Théorème 3.3.1 *Le problème du sac-à-dos est NP-difficile*[1].

1. Nous reviendrons à la fin du chapitre sur la notion des problèmes NP-difficiles.

Le théorème est admis sans démonstration.
Une façon commode de résoudre ce problème, si les p_i sont entiers (ce à quoi on peut toujours se ramener en changeant l'unité de mesure) et si P n'est pas trop grand, passe par l'utilisation de la programmation dynamique (qui n'est pas abordée ici), laquelle fournit un algorithme pseudo-polynomial en $O(nP)$. Nous allons présenter d'autres façons de résoudre ce problème.

3.3.1.1 Méthode approchée

Une méthode approchée a pour but de trouver une solution avec un bon compromis entre la qualité de la solution et le temps de calcul. Pour le problème du sac à dos, voici un exemple d'algorithme de ce type :

— calculer le rapport (v_i/p_i) pour chaque objet i ;

— trier tous les objets par ordre décroissant de cette valeur ;

— sélectionner les objets un à un dans l'ordre du tri et ajouter l'objet sélectionné dans le sac si le poids maximal reste respecté.

Déroulons cet algorithme sur notre exemple :
Première étape :

Objets	1	2	3	4
v_i/p_i	0,54	0,33	0,37	0,30

Deuxième étape :

Objets	1	3	2	4
v_i	7	3	4	3
p_i	13	8	12	10
v_i/p_i	0,54	0,37	0,33	0,30

Troisième étape :
Objet 1 : on le met dans le sac vide, le poids du sac est alors de 13 et inférieur à 30 ;
Objet 3 : on le met dans le sac car $13 + 8 = 21$ est inférieur à 30 ;
Objet 2 : on ne le met pas dans le sac car le poids total (33) dépasserait 30.
Objet 4 : on ne le met pas dans le sac (poids total de 31).
La solution trouvée est donc de mettre les objets 1 et 3 dans le sac, ce qui donne une valeur de 10. Cette solution n'est pas optimale, puisqu'une solution avec une valeur totale de 11 existe. Néanmoins, cet algorithme reste rapide même si le nombre d'objets augmente considérablement.
Ce type d'algorithme est aussi appelé algorithme glouton, car il ne remet jamais en cause une décision prise auparavant. Ici, lorsque l'objet 2 ne peut pas entrer dans le sac, l'algorithme n'essaie pas d'enlever l'objet 3 du sac pour y mettre l'objet 2 à sa place.

3.3.1.2 Formulation sous forme d'un programme linéaire et branch-and-bound

Programmation linéaire On peut modéliser le problème du sac-à-dos sous forme d'un programme linéaire en nombres entiers.

$$\begin{array}{ll} \text{Max} & \sum_{j=1}^{n} c_j x_j \\ \text{s.c.} & \sum_{j=1}^{n} p_j x_j \leq P \\ & x_j \in \{0,1\} \quad j \in \{1,\ldots,n\}. \end{array}$$

Remarquons que dans un sens ce programme linéaire en nombres entiers est le plus simple possible : chaque variable ne peut prendre que deux valeurs, et il n'y a qu'une seule contrainte. La relaxation continue fournit une borne supérieure naturelle à la solution du programme précédent

$$\begin{array}{ll} \text{Max} & \sum_{j=1}^{n} c_j x_j \\ \text{s.c.} & \sum_{j=1}^{n} p_j x_j \leq P \qquad j \in \{1, \ldots, n\}. \\ & 0 \leq x_j \leq 1 \end{array}$$

Pour la calculer (pour faire du *branch-and-bound* par exemple - voir ci-dessous), on pourrait bien sûr faire appel à l'algorithme du simplexe ou à l'algorithme des points intérieurs. Mais il existe un algorithme très simple, glouton, qui calcule la solution optimale du relâché continu.
On suppose que $\sum_{j=1}^{n} p_j > P$, sinon, le problème du asc-à-dos est trivial. Ensuite on fait
Classer les objets de façon à ce que

$$\frac{c_1}{p_1} \geq \frac{c_2}{p_2} \geq \ldots \geq \frac{c_n}{p_n}.$$

Poser $x_1 = x_2 = \ldots = x_{\bar{j}} := 1$ avec $\bar{j}$ le plus grand entier tel que $\sum_{j=1}^{\bar{j}} p_j \leq W$.
Poser $x_{\bar{j}+1} := \frac{P - \sum_{j=1}^{\bar{j}} p_j}{p_{\bar{j}+1}}$.
Poser $x_{\bar{j}+2} = \cdots = x_n := 0$.

Lemme 3.3.2 *Un tel $\boldsymbol{x}$ est solution optimale du relâché continu.*

Elément de preuve. - On prouve d'abord que si $\boldsymbol{x}$ est optimal avec $x_i < 1, x_k > 0$ et $i < k$, alors $c_i/p_i = c_k/p_k$. Par conséquent, il existe une solution optimale avec un indice $\bar{j}$ tel que $x_i = 1$ pour tout $i < \bar{j}$ (si $\bar{j} \geq 2$) et tel que $x_i = 0$ pour tout $i > \bar{j}$ (si $\bar{j} \leq n-1$). □
On sait donc calculer une borne d'assez bonne qualité (relâché continu) en $O(n \log(n))$. Cette borne permet de mettre en place un branch-and-bound. Pour le branchement, c'est la technique usuelle pour les programmes linéaires en $\{0, 1\}$: fixer certaines variables à 0 et d'autres à 1.

Exemple 3.3.3 *Toute formulation de ce problème commence par un énoncé des données. Dans notre cas, nous avons un sac à dos de poids maximal P et n objets. Pour chaque objet i, nous avons un poids p_i et une valeur v_i. Pour quatre objets ($n = 4$) et un sac à dos d'un poids maximal de* 30 kg*($P = 30$), nous avons par exemple les données suivantes :*

Objets	*1*	*2*	*3*	*4*
v_i	*7*	*4*	*3*	*3*
p_i	*13*	*12*	*8*	*10*

Ensuite, il nous faut définir les variables qui représentent en quelque sorte les actions ou les décisions qui amèneront à trouver une solution. On définit la variable x_i associée à un objet i de la façon suivante : $x_i =$ 1si l'objet i est mis dans le sac, et $x_i =$ 0si l'objet i n'est pas mis dans le sac.
Dans notre exemple, une solution réalisable est de mettre tous les objets dans le sac à dos sauf le premier, nous avons donc : $x_1 = 0, x_2 = 1, x_3 = 1$, et $x_4 = 1$.
Puis il faut définir les contraintes du problème. Ici, il n'y en a qu'une : la somme des poids de tous les objets dans le sac doit être inférieure ou égale au poids maximal du sac à dos.
Cela s'écrit ici $x_1 \cdot p_1 + x_2 \cdot p_2 + x_3 \cdot p_3 + x_4 \cdot p_4 \leq P$ et pour n objets :

$$\sum_{i=1}^{i=n} x_i \cdot p_i \leq P$$

Pour vérifier que la contrainte est respectée dans notre exemple, il suffit de calculer cette somme : $0 \times 13 + 1 \times 12 + 1 \times 8 + 1 \times 10 = 30$, ce qui est bien inférieur ou égal à 30, donc la contrainte est respectée. Nous parlons alors de solution réalisable. Mais ce n'est pas nécessairement la meilleure solution.
Enfin, il faut exprimer la fonction qui traduit notre objectif : maximiser la valeur totale des objets dans le sac. Pour n objets, cela s'écrit :

$$\max \sum_{i=1}^{i=n} x_i \cdot v_i$$

Dans notre exemple, la valeur totale contenue dans le sac est égale à 10 . Cette solution n'est pas la meilleure, car il existe une autre solution de valeur plus grande que 10 : il faut prendre seulement les objets 1 et 2 qui donneront une valeur totale de 11. Il n'existe pas de meilleure solution que cette dernière, nous dirons alors que cette solution est optimale.

Branch-and-bound ou procédures par séparation et évaluation (PSE) Pour trouver la solution optimale, et être certain qu'il n'y a pas mieux, il faut utiliser une méthode exacte, qui demande un temps de calcul beaucoup plus long (si le problème est difficile à résoudre). Il n'existe pas une méthode exacte universellement plus rapide que toutes les autres. Chaque problème possède des méthodes mieux adaptées que d'autres. Nous allons présenter un exemple d'algorithme de ce type, nommé procédure par séparation et évaluation (PSE), ou en anglais branch and bound. Nous n'exposerons ici qu'une version simplifiée d'une PSE.
Une PSE est un algorithme qui permet d'énumérer intelligemment toutes les solutions possibles. En pratique, seules les solutions potentiellement de bonne qualité seront énumérées, les solutions ne pouvant pas conduire à améliorer la solution courante ne sont pas explorées.
Pour représenter une PSE, nous utilisons un "arbre de recherche" constitué :

— de nœuds ou sommets, où un nœud représente une étape de construction de la solution
— d'arcs pour indiquer certains choix faits pour construire la solution.

Dans notre exemple, les nœuds représentent une étape pour laquelle des objets auront été mis dans le sac, d'autres auront été laissés en dehors du sac, et d'autres, encore, pour lesquels aucune décision n'aura encore été prise. Chaque arc indique l'action de mettre un objet dans le sac ou, au contraire, de ne pas le mettre dans le sac. La figure suivante représente l'arbre de recherche du problème donné en exemple.
On associe l'ensemble vide à la racine (dont la profondeur est 0) et pour un nœud de profondeur $i-1$, on construit deux fils : celui du haut où l'on ajoute l'objet i dans le sac et celui du bas où le sac reste tel quel. L'arbre de recherche achevé, chaque feuille représente alors une solution potentielle mais pas forcément réalisable. Dans le schéma, les feuilles au bord épais représentent les propositions irréalisables car supérieures au poids maximal à ne pas
dépasser. Pour déterminer la solution, il suffit de calculer la valeur du sac pour chaque nœud feuille acceptable et de prendre la solution ayant la plus grande valeur.
Cependant la taille de l'arbre de recherche est exponentielle en le nombre d'objets. Aussi il existe de nombreuses techniques algorithmiques de parcours de ce type d'arbre. Ces techniques ont pour but de diminuer la taille de l'arbre et d'augmenter la rapidité du calcul. Par exemple, on peut remarquer que le poids du nœud interne $\{1, 2, 3\}$ dépasse déjà le poids maximal, il n'était donc pas nécessaire de développer l'étape suivante avec l'objet 4 . Les procédures par séparation et évaluation (PSE) permettent d'élaguer encore plus cet arbre en utilisant des bornes inférieures et supérieures de la fonction objectif :

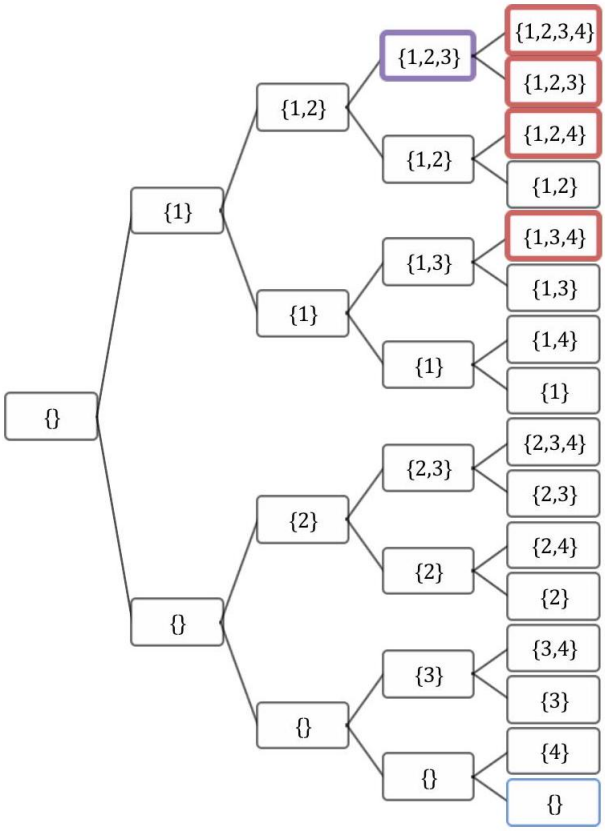


FIGURE 3.3 – Arbre de solution par branch-and-bound.

— Une borne inférieure est une valeur minimum de la fonction objectif. Autrement dit, c'est une valeur qui est nécessairement inférieure à la valeur de la meilleure solution possible. Dans notre cas, une configuration du sac réalisable quelconque fournit une borne inférieure.

— Une borne supérieure est une valeur maximale de la fonction objectif. Autrement dit, nous savons que la meilleure solution a nécessairement une valeur plus petite. Dans notre cas, nous savons que la valeur de la meilleure solution est nécessairement inférieure à la somme des valeurs de tous les objets (comme si on pouvait tous les mettre dans le sac).

Supposons maintenant que la borne inférieure soit initialisée par l'algorithme heuristique vu précédemment. Pendant la recherche à chaque nœud, nous pouvons déterminer une borne supérieure : la somme de toutes les valeurs de tous les objets déjà mis dans le sac plus la somme des valeurs des objets restants dont on ne sait pas encore s'ils seront dans le sac. À partir d'un nœud et de sa borne supérieure, nous savons que les solutions descendantes de ce nœud ne pourront pas avoir une valeur plus grande que cette borne supérieure. Si jamais, à un nœud donné, la valeur de la borne supérieure est inférieure à la valeur de la borne inférieure, alors il est inutile d'explorer les nœuds descendants de celui-ci. On dit qu'on coupe l'arbre de recherche. En effet, si à partir d'un nœud, nous savons que nous ne pourrons pas faire plus de 10 (borne supérieure calculée) et que la borne inférieure existante est à 11 (on a déjà une solution de valeur 11), alors les solutions descendantes de ce nœud ne sont pas intéressantes. Enfin, dernière remarque, la valeur de la borne inférieure peut être actualisée lorsqu'est trouvée une solution réalisable qui possède une valeur plus grande.
Ce système de calcul de bornes demande un faible coût de temps de calcul, et permet d'augmenter la rapidité de la PSE puisqu'elle coupe des branches d'arbre pour ne pas perdre de temps à les explorer. D'autres techniques servent à diminuer la taille de l'arbre et à augmenter la rapidité. Par exemple,

elles sont basées sur l'ordre dans lequel on prend les décisions sur les objets, ou sur une évaluation à chaque nœud, ou encore sur des propriétés du problème qui permettent de déduire des conclusions sur certains objets. Sur notre exemple initial, il est possible d'obtenir de façon exacte la solution optimale en n'examinant que 9 nœuds au lieu de 31.

3.3.2 Problème du bin-packing

Le problème du bin-packing traite du cas où l'on a des objets de tailles variables et un seul type de conteneurs, et où l'on se demande comment utiliser un nombre minimum de conteneurs pour ranger tous les objets. Dans les formes les plus générales de ce problème, on peut aussi prendre en compte la forme des objets (on parle alors de bin-packing 2 D ou de bin-packing 3D), des incompatibilités, etc. Ce problème peut être parfois appelé aussi cutting-stock. En effet, les problèmes de découpes (de pièces de textile, métal, etc.), où l'on cherche à minimiser les pertes, se modélisent de façon similaire. Ici, on se limite au cas le plus simple, 1D. C'est un cas déjà très utile en pratique, puisqu'il peut fournir des bornes, être utilisé en sous-routines, ou être appliqué tel quel (nombre min de CD-rom pour stocker le contenu d'un disque dur, découper des planches de longueurs variables dans des grandes planches de longueur fixée, découpe dans des bandes de tissu, etc.).
Le problème s'écrit alors formellement :
Donnée : Des entiers positifs ou nuls $a_1, \ldots, a_n, W$.
Tâche : Trouver un entier naturel k minimium et une affectation $f : \{1, \ldots, n\} \to \{1, \ldots, k\}$ avec $\sum_{i:f(i)=j} a_i \leq 1$ pour tout $j \in \{1, \ldots, k\}$.

Théorème 3.3.4 *Le problème du bin-packing est NP-difficile.*

Non seulement il est NP-difficile, mais il existe encore de nombreuses instances de "petite" taille non résolue, ce qui justifie et motive largement des travaux de recherche dans ces domaines. Par exemple, considérons l'instance suivante, dont on ne connaît pas à ce jour la solution optimale.
Taille de la boîte : 150
Nombre d'objets : 120
Taille des objets : 100 22 25 51 95 58 97 30 79 23 53 80 20 65 64 21 26 100 81 98 70 85 92 97 86 71 91 29 63 34 67 23 33 89 94 47 100 37 40 58 73 39 49 79 54 57 98 69 674 93 83 49 62 792 82 69 45 69 20 75 97 51 70 29 91 98 77 48 45 43 61 36 82 89 94 26 35 58 58 57 46 44 91 49 52 65 42 33 60 37 57 91 52 95 84 72 75 89 81 67 74 87 60 32 76 85 59 62 39 64 52 88 45 29 88 85 54 40 57
meilleure solution connue : 51 boîtes.

Le défi est de trouver une solution en moins de 51 boîtes, ou alors de parvenir à montrer que 51 boîtes est la solution optimale. Nous allons décrire maintenant quelques heuristiques classiques (NEXT-FIT, FIRST-FIT, FIRST-FIT DEACRISING), puis nous verrons les approches branch-and-bound.

3.3.2.1 Quelques heuristiques classiques.

NEXT-FIT. - Je prends les objets les uns après les autres. Dès que l'objet i ne peut pas entrer dans la boîte courante, je passe à une nouvelle boîte. De façon plus formelle :

1.

1. source : la page web du professeur Eric Taillard http ://mistic.heig-vd.ch/taillard/

Commencer par $k := 1, i := 1$ et $S := 0$. Répéter

> Si $S + a_i > 1$, faire $k := k + 1$ et $S := 0$. Sinon, faire $f(i) := k, S := S + a_i$ et $i := i + 1$.

On a

Théorème 3.3.5 *NEXT-FIT fournit une solution SOL telle que*

$$SOL \leq 2OPT - 1,$$

où OPT est la valeur d'une solution optimale

La démonstration et toutes celles qui suivent dans ce paragraphes seront omises. Le lecteur intéressé peut les retrouver dans [4] aux pages 91 à 94.

FIRST-FIT. - Je prends les objets les uns après les autres. Je mets l'objet i dans la boîte de plus petit rang où il peut entrer. De façon plus formelle,
poser $i := 1$. Répéter :

$$\text{poser } f(i) := \min \left\{ j \in \mathbb{N} : a_i + \sum_{h<i : f(h)=j} a_h \leq W \right\} ; \text{ poser } i := i + 1.$$

Poser $k := \max_{i \in \{1,\ldots,n\}} f(i)$.

Théorème 3.3.6 *FIRST-FIT fournit une solution SOL telle que*

$$SOL \leq \left\lceil \frac{17}{10} OPT \right\rceil$$

où OPT est la valeur d'une solution optimale.

La preuve de ce résultat, difficile, est omise. Ce que ce résultat indique et qui est vérifié en pratique, c'est que l'heuristique FIRST-FIT marche en général mieux que l'heuristique NEXTFIT

FIRST-FIT DECREASING. - Je trie d'abord les objets par a_i décroissant. Puis j'applique FIRST-FIT.

Théorème 3.3.7 *FIRST-FIT DECREASING fournit une solution SOL telle que*

$$SOL \leq \frac{3}{2} OPT.$$

C'est cette heuristique qui a en général les meilleurs résultats. L'avantage des deux premiers algorithmes sur ce dernier est qu'ils peuvent fonctionner on-line, ce qui signifie qu'on peut les faire tourner lorsque les objets arrivent les uns après les autres et qu'on ne connaît pas la taille des objets futurs. On peut aussi utiliser le branch-and-bound mais cela nécessite la relaxation continue ou la relaxation lagragienne que nous n'avons pas abordées ici. On peut néamoins en avoir l'idée en lisant [4].

3.4 Problème du voyageur de commerce

Le problème du voyageur de commerce (TSP, *Travelling Salesman Problem*) est un problème classique en optimisation combinatoire et en théorie des graphes. Il peut être formulé ainsi : étant donné un ensemble de n villes et les coûts ou distances entre chaque paire de villes, trouver le cycle Hamiltonien de coût minimal qui visite chaque ville exactement une fois et revient à la ville de départ. Nous donnons une equisse du problème par manque de temps mais un lecteur intéressé peut consulter les nombreuses sources sur ce problème dont [8, 4].

Formellement, soit un graphe complet pondéré $G = (V, E)$, avec $V = \{v_1, v_2, \ldots, v_n\}$ et des poids c_{ij} pour chaque arête $(v_i, v_j) \in E$. Le TSP consiste à résoudre :

$$\min_{\pi \in S_n} \sum_{k=1}^{n} c_{\pi(k)\pi(k+1)}, \quad \pi(n+1) = \pi(1)$$

où S_n est l'ensemble des permutations de $\{1, 2, \ldots, n\}$.

Le TSP est un problème NP-difficile. Il n'existe pas dalgorithme polynomial connu pour le résoudre exactement. On distingue deux catégories de méthodes :

- **Méthodes exactes** : branch-and-bound, programmation linéaire, algorithmes de type Held-Karp.
- **Méthodes heuristiques et métaheuristiques** : algorithmes gloutons, 2-opt, 3-opt, recuit simulé, algorithmes génétiques.

3.4.1 Résultats théoriques

Définition 3.4.1 (Cycle Hamiltonien) *Un cycle Hamiltonien dans un graphe $G = (V, E)$ est un cycle qui passe exactement une fois par chaque sommet de V.*

Définition 3.4.2 (TSP symétrique et asymétrique)
- *Symétrique (STSP) : $c_{ij} = c_{ji}$ pour toutes les paires i, j.*
- *Asymétrique (ATSP) : $c_{ij} \neq c_{ji}$ possible.*

Théorème 3.4.3 (Complexité du TSP) *Le problème du voyageur de commerce est NP-difficile. Plus précisément, le problème de décision : "Existe-t-il un cycle Hamiltonien de coût $\leq K$?" est NP-complet.*

Lemme 3.4.4 (Inégalité triangulaire) *Si les distances respectent l'inégalité triangulaire : $c_{ik} \leq c_{ij} + c_{jk}$ pour tout i, j, k, alors certaines heuristiques comme le nearest neighbor ont une approximation bornée.*

3.4.2 Programmation linéaire pour le TSP

Le TSP peut être formulé en PL :

Définition 3.4.5 (Formulation en variables binaires) *Soit $x_{ij} = 1$ si larête (i, j) est utilisée dans le cycle, 0 sinon.*

$$\begin{aligned} \min \quad & \sum_{i<j} c_{ij} x_{ij} \\ s.c. \quad & \sum_{j \neq i} x_{ij} = 2 \quad \forall i \in V \\ & \sum_{i,j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subset V, 2 \leq |S| \leq n-1 \\ & x_{ij} \in \{0, 1\} \end{aligned}$$

Proposition 3.4.6 (Relaxation linéaire) *En relaxant $x_{ij} \in [0, 1]$, on obtient une borne inférieure sur le coût minimal.*

3.4.3 Algorithmes classiques

Algorithm 9 Algorithme glouton (Nearest Neighbor)

1. Choisir une ville de départ v_0.
2. Tant que toutes les villes ne sont pas visitées :
   (a) Choisir la ville non visitée la plus proche de la dernière ville visitée.
   (b) Ajouter cette ville au chemin.
3. Retourner à v_0 pour fermer le cycle.

Algorithm 10 Algorithme 2-opt

1. Commencer avec un cycle initial (par exemple via nearest neighbor).
2. Tant quil existe une amélioration possible :
   (a) Sélectionner deux arêtes non adjacentes et les échanger si le coût total diminue.
3. Répéter jusquà ce quaucune amélioration ne soit possible.

Exemple 3.4.7 (TSP à 4 villes) *Considérons les villes A, B, C, D avec la matrice des distances suivante :*

$$\begin{bmatrix} 0 & 10 & 15 & 20 \\ 10 & 0 & 35 & 25 \\ 15 & 35 & 0 & 30 \\ 20 & 25 & 30 & 0 \end{bmatrix}$$

Solution exacte : *On peut énumérer tous les cycles Hamiltoniens et calculer les coûts :*

$$\begin{aligned} A \to B \to C \to D \to A &= 10 + 35 + 30 + 20 = 95 \\ A \to B \to D \to C \to A &= 10 + 25 + 30 + 15 = 80 \\ A \to C \to B \to D \to A &= 15 + 35 + 25 + 20 = 95 \\ A \to C \to D \to B \to A &= 15 + 30 + 25 + 10 = 80 \\ A \to D \to B \to C \to A &= 20 + 25 + 35 + 15 = 95 \\ A \to D \to C \to B \to A &= 20 + 30 + 35 + 10 = 95 \end{aligned}$$

Le cycle minimal est donc $A \to B \to D \to C \to A$ *avec coût* 80.

Exemple 3.4.8 (TSP à 5 villes heuristique) *Soit les villes* $V = \{1, 2, 3, 4, 5\}$ *avec distances :*

$$\begin{bmatrix} 0 & 2 & 9 & 10 & 7 \\ 2 & 0 & 6 & 4 & 3 \\ 9 & 6 & 0 & 8 & 5 \\ 10 & 4 & 8 & 0 & 6 \\ 7 & 3 & 5 & 6 & 0 \end{bmatrix}$$

Algorithme nearest neighbor : *départ de la ville 1*

$$1 \to 2 \to 5 \to 3 \to 4 \to 1$$

Coût total : $2 + 3 + 5 + 8 + 10 = 28$

On peut ensuite appliquer 2-opt pour améliorer le chemin. Supposons quun échange réduit le coût à 26, donnant un cycle approché optimal. Cette méthode illustre lusage combiné de heuristiques pour résoudre des instances de taille plus grande.

3.4.4 Problèmes ouverts (de recherche) sur le voyageur de commerce

Malgré de nombreuses décennies de recherche, le TSP présente encore plusieurs problèmes ouverts :

1. **Algorithme polynomial exact ?** Existe-t-il un algorithme polynomial qui résout le TSP exact pour toutes les instances générales ? La conjecture actuelle est que cela est impossible, mais une preuve formelle liée à P vs NP nexiste pas encore.
2. **Bornes dapproximation pour TSP asymétrique** Pour le TSP symétrique avec inégalité triangulaire, il existe des approximations polynomiales efficaces (algorithme de Christofides). Pour le TSP asymétrique, trouver un algorithme dapproximation avec un facteur constant reste un problème ouvert majeur.
3. **Étude des instances structurellement difficiles** Identifier les classes dinstances où les heuristiques échouent systématiquement ou produire des instances ń difficiles ż pour les solveurs reste un domaine actif de recherche.
4. **Intégration avec dautres contraintes** Le TSP combiné avec des contraintes supplémentaires (fenêtres temporelles, capacités, multivehicles) pose des défis théoriques et pratiques importants.
5. **Approches probabilistes et aléatoires** Comprendre la distribution des coûts des cycles dans les grands graphes aléatoires et établir des limites probabilistes rigoureuses constitue un problème ouvert en mathématiques appliquées et optimisation.

3.5 Comment devenir millionaire : les problèmes NP-difficiles

3.5.1 Algorithme et complexité

Algoritme, algorithme exact, algorithme approché. Une fois que l'on a modélisé notre problème concret en un problème formalisé ou en un programme mathématique, on doit se demander

comment le résoudre. Dans ce cours, nous verrons différents problèmes, différents programmes mathématiques, et quelles sont les méthodes pour les résoudre. De nouveaux problèmes sont proposés tout le temps par les chercheurs travaillant en recherche opérationnelle ; dans ce cours, nous nous limiterons bien entendu aux plus classiques.
On se fixe un ensemble d'opérations que l'on considère ńfacile à faireż. Par exemple, comparaison d'entiers, lire une adresse mémoire, etc. Une suite d'opérations élémentaires permettant de résoudre un problème s'appelle un algorithme. La résolution d'un problème de recherche opérationnelle passe toujours par l'application d'un algorithme, qui est ensuite implémenté. Si le problème que l'on tente de résoudre est un programme d'optimisation, on parlera d'algorithme exact si l'algorithme est sûr de se terminer avec l'optimum du programme, et d'algorithme approché sinon. La question qui se pose également est celle de l'efficacité de l'algorithme, i.e. du temps qu'il va mettre pour résoudre le problème (une fois admis que l'algorithme est correct et résout bien le problème).

Question de l'efficacité. Une méthode peut être de tester l'algorithme sur de grandes quantités de données. Mais ces tests-là peuvent être très, très longs, et de toute façon, à chaque fois que l'on va penser à un algorithme possible, on ne va pas systématiquement procéder à son implémentation et à des tests. La théorie de la complexité, un des fondement théorique de la recherche opérationnelle, a pour but de mesurer a priori l'efficacité d'un algorithme. La petite histoire qui va suivre va montrer l'intérêt de cette théorie, et est librement adapté du livre de Garey et Johnson [10].

De l'intérêt de la théorie pour sauver son boulot. Supposez qu'un jour votre patron vous appelle dans son bureau et vous annonce que l'entreprise est sur le point d'entrer sur le marché compétitif du schmilblick. Il faut donc trouver une bonne méthode qui permette de dire si une collection de spécifications peuvent être satisfaites ou non, et si oui, qui donne la façon de construire ce schmilblick. On peut imaginer qu'un schmilblick est décrit par des variables booléennes C_i qui indiquent si le composant i est présent ou non. De plus, on a des règles du type si C_1 et C_5 sont vraies et C_3 est faux, alors C_7 doit être faux. Etc.
Comme vous travaillez au département de Recherche Opérationnelle, c'est à vous de trouver un algorithme efficace qui fasse cela.
Après avoir discuté avec le département de production des schmilblicks afin de déterminer quel est exactement le problème à résoudre, vous retournez dans votre bureau et commencez avec enthousiasme à travailler d'arrache-pied. Malheureusement, quelques semaines plus tard, vous êtes obligé de l'admettre : vous n'avez pas réussi à trouver un algorithme qui fasse substantiellement mieux que d'essayer toutes les possibilités - ce qui ne réjouira certainement pas votre patron puisque cela demandera des années et des années pour tester juste un ensemble de spécifications. Vous ne souhaitez certainement pas retourner dans le bureau de votre patron et lui dire : "Je ne parviens pas à trouver d'algorithme efficace. Je pense que je suis trop bête."
Ce qui serait bien, c'est que vous soyez capable de démontrer que le problème du schmilblick est intrinséquement intractable [(3)]. Dans ce cas, vous pourriez sereinement aller voir votre patron
3. anglicisme signifiant dans ce contexte qu'il n'existe pas d'algorithme efficace résolvant le problème considéré.
et lui dire : "Je ne parviens pas à trouver d'algorithme efficace car un tel algorithme ne peut exister."
Malheureusement, personne à ce jour n'a été capable de montrer qu'il existe un problème intrinsèquement intractable. En revanche, les théoriciens de l'informatique ont développé un concept qui est presqu'aussi bon, celui de problème NP-complet. La théorie de la NP-complétude fournit des

techniques variées pour prouver qu'un problème est aussi difficile qu'une grande quantité d'autres problèmes, pour lesquels aucune méthode efficace n'a été trouvée à ce jour, malgré les efforts répétés des plus grands experts de la planète. Avec ces techniques, vous pourrez peut-être prouver que votre problème est NP-complet et donc équivalent à tous ces problèmes difficiles. Vous pourriez alors entrer dans le bureau de votre patron et annoncer : "Je ne parviens pas à trouver d'algorithme efficace, mais aucune star de la RO, aucune star de l'informatique théorique, aucune star de l'optimisation discrète ne peut le faire." Au pire, cela l'informera qu'il ne sert à rien de vous virer pour embaucher un autre expert à votre place.
Prouver qu'un problème est NP-complet ne signifie pas la fin de l'histoire ; au contraire, c'est le début du vrai travail. Le fait de savoir qu'un problème est NP-complet fournit une information sur l'approche la plus productive. Cela montre qu'il ne faut pas se focaliser sur la recherche d'un algorithme exact et efficace et qu'il faut avoir des approches moins ambitieuses. Par exemple, on peut chercher des algorithmes efficaces résolvant divers cas particuliers. On peut chercher des algorithmes sans garantie sur le temps d'exécution, mais qui en général semblent être rapides. Ou alors, on peut ń relaxer ż le problème et chercher un algorithme rapide qui trouve des schmilblicks satisfaisant presque toutes les spécifications. Dans les chapitres suivants, nous verrons des exemples concrets illustrant de telles approches.

3.5.2 Notions de base en théorie de la complexité ou comment gagner 1 million

de $. - Pour évaluer théoriquement l'efficacité d'un algorithme résolvant un problème $\mathcal{P}$, on compte le nombre d'opérations élémentaires que cet algorithme effectue pour le résoudre. Comme la taille des données de $\mathcal{P}$ peut varier, on va avoir une fonction de complexité f qui à n, la taille des données, va associer le nombre $f(n)$ d'opérations élémentaires que l'algorithme va effectuer pour trouver une solution. La taille des données, c'est le nombre de bits qu'il faut pour coder les données.
Rappelons qu'en mathématiques, dire qu'une fonction f est $O(g(n))$ (lire ńgrand 'o' de $g(n)$»), c'est dire qu'il existe une constante B telle que $f(n) \leq Bg(n)$ pour tout n. Si la fonction de complexité d'un algorithme est un $O(p(n))$, où p est un polynôme, alors l'algorithme est dit polynomial. Sinon, il est dit exponentiel. Un algorithme polynomial est en général perçu comme efficace, un algorithme exponentiel est en général mauvais. Le Tableau 3.7 montre le temps qu'il faut à des algorithmes de fonction de complexité variable pour résoudre un problème pour différentes tailles de données.
On entend souvent dire : ńCes problèmes ne se poseront plus lorsque la puissance des ordinateurs aura augmenté ≫. C'est faux, comme le montre le Tableau 3.8. Supposons par exemple que l'on ait un algorithme résolvant le problème du schmilblick $B2^n$ opérations élémentaires, où B est une constante et où n est le nombre de spécifications (avec donc un ordinateur effectuant bien plus d'1 milliard d'opérations par seconde). Et supposons que l'on soit capable de résoudre des problèmes de schmilblick en 1 heure jusqu'à $n = 438$. L'utilisation d'une machine 1000 fois plus rapide ne permettra que d'aller jusqu'à $n = 448$ en 1 heure !
Un problème $\mathcal{P}$ est dite polynomial ou dans **P** s'il existe un algorithme polynomial qui le résout. Taille de l'instance la plus large que l'on peut résoudre en 1 heure Un problème de décision $\mathcal{P}$ est dans NP si, lorsque la réponse est ńouiż, il existe un certificat (c'est-à-dire une information supplémentaire) et un algorithme polynomial qui permet de vérifier que la réponse est ńouiż, sans pour autant être capable de trouver cette réponse (voir des exemples ci-dessous). Le sigle NP ne signifie pas ń non polynomial ż mais ń non déterministiquement polynomial ». Le non-déterminisme ici fait référence aux machines de Turing non-déterministes, et dépasse largement le cadre de ce cours.
Un problème de décision est dit NP-complet si l'existence d'un algorithme polynomial le résolvant

	Taille n					
Complexité	10	20	30	40	50	60
n	$0,01\mu$ s	$0,02\mu$ s	$0,03\mu$ s	$0,04\mu$ s	$0,05\mu$ s	$0,06\mu$ s
n^2	$0,1\mu$ s	$0,4\mu$ s	$0,9\mu$ s	$1,6\mu$ s	$2,5\mu$ s	$3,6\mu$ s
n^3	1μ s	8μ s	27μ s	64μ s	125μ s	216μ s
n^5	0,1 ms	3,2 ms	24,3 ms	102,4 ms	312,5 ms	777,6 ms
2^n	$\sim 1\mu$ s	~ 1 ms	~ 1 s	~ 18 min20 s	~ 13 jours	~ 36 ans 6 mois

TABLE 3.7 – Comparaison de diverses fonctions de complexité pour un ordinateur effectuant 1 milliard d'opérations par seconde.

Fonction de complexité	Avec un ordinateur actuel	Avec un ordinateur 100 fois plus rapide	Avec un ordinateur 1000 fois plus rapide
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31.6N_2$
n^3	N_3	$4.64N_3$	$10N_3$
n^5	N_4	$2.5N_4$	$3.98N_4$
2^n	N_5	$N_5 + 6.64$	$N_5 + 9.97$
3^n	N_6	$N_6 + 4.19$	$N_6 + 6.29$

TABLE 3.8 – Comparaison de diverses fonctions de complexité.

implique l'existence d'un algorithme polynomial pour tout problème NP. A ce jour, on ne connaît pas d'algorithme polynomial résolvant un problème NP-complet. En revanche, on connaît beaucoup de problèmes NP-complets. Le premier a été trouvé en 1970, par un informaticien appelé Cook. Considérons le problème suivant.

3.5.2.1 Exemples de problèmes NP-difficiles : cycle eulérien et cycle hamiltonien

⊗ Donnée : Un graphe $G = (V, E)$.
Question : G a-t-il un cycle eulérien ?
Ce problème de décision est dans **P**. En effet, le Théorème 2.2.1 ci-dessus permet facilement de répondre à la question en temps polynomial : si m est le nombre d'arêtes de G et n son nombre de sommets, tester le fait que tous les sommets sont de degré pair prend $O(m)$ et que le graphe est connexe prend $O(n + m)$, au total $O(n + m)$. Il est donc également dans NP : le certificat est le graphe lui-même.

De même, le problème suivant est dans **P**.

Problème de la chaîne eulérienne Donnée : Un graphe $G = (V, E)$ et deux sommets $s, t \in V$.
Question : G a-t-il une $s - t$ chaîne eulérienne ?
Considérons maintenant le problème suivant.

Problème du cycle hamiltonien Donnée : Un graphe $G = (V, E)$.
Question : G a-t-il un cycle hamiltonien ?
A ce jour, nul n'a pu démontrer que ce problème est dans **P**, ni qu'il n'y était pas. En revanche,

il est facile de voir que ce problème est dans NP car si la réponse est positive, alors l'algorithme qui consiste à suivre le cycle hamiltonien permet de prouver que la réponse est bien positive. Ici, le cycle hamiltonien joue le rôle de certificat. Il a été également démontré que ce problème est NP-complet. Cela signifie que si vous rencontrez quelqu'un vous disant qu'il connaît un algorithme efficace pour répondre à cette question, il est très probable qu'il se trompe car tous les problèmes NP-complets pourraient alors être résolus par un algorithme polynomial. Or personne à ce jour n'a pu en trouver, et quantité de gens très brillants ont cherché un tel algorithme. S'il ne se trompe pas, c'est une découverte fondamentale, qui aurait un impact énorme tant dans le monde de l'informatique théorique, que dans la recherche opérationnelle appliquée. De plus, cette personne percevrait le prix d'1 millions de dollars offert par la Fondation Clay pour la résolution de la question ouverte $\mathbf{P} \stackrel{?}{=} \text{NP}$. De même le problème suivant est NP-complet.

Problème de la chaîne hamiltonienne Donnée : Un graphe $G = (V, E)$, deux sommets $s, t \in V$. Question : G a-t-il une $s - t$ chaîne hamiltonienne ?
On peut définir également les mêmes problèmes pour les graphes orientés, les résultats seront semblables. En résumé, on a

Résultats de complexité pour des parcours.	
Problèmes	Complexité
Graphe non-orienté : Cycle eulérien	P
Graphe non-orienté : Chaîne eulérienne	P
Graphe non-orienté : Cycle hamiltonien	NP-complet
Graphe non-orienté : Chaîne hamiltonienne	NP-complet
Graphe orienté : Circuit eulérien	P
Graphe orienté : Chemin eulérien	**P**
Graphe orienté : Circuit hamiltonien	NP-complet
Graphe orienté : Chemin hamiltonien	NP-complet

La Figure 3.4 illustre les relations entre problèmes **P**, NP, NP-complets, si $\mathbf{P} \neq \mathbf{NP}$ (ce qui est le plus probable).
La notion de problème NP et a fortiori de problème NP-complet est définie pour les problèmes de décision, dont la réponse est ńouiż ou ńnon ż. Mais les problèmes que l'on rencontre en RO sont rarement de ce type. Il existe une notion qui permet de caractériser des problèmes difficiles qui ne sont pas des problèmes de décision : c'est celle de problème NP-difficile. Un problème est NP-difficile si savoir le résoudre en temps polynomial impliquerait que l'on sait résoudre un problème (de décision) NP-complet en temps polynomial. Les problèmes NPdifficiles sont donc dans un sens plus dur que les problèmes NP-complets. Par exemple :

Théorème 3.5.1 *Le problème du voyageur de commerce est NP-difficile.*

La preuve de ce résultat découle de la NP-complétude du problème de cycle hamiltonien.

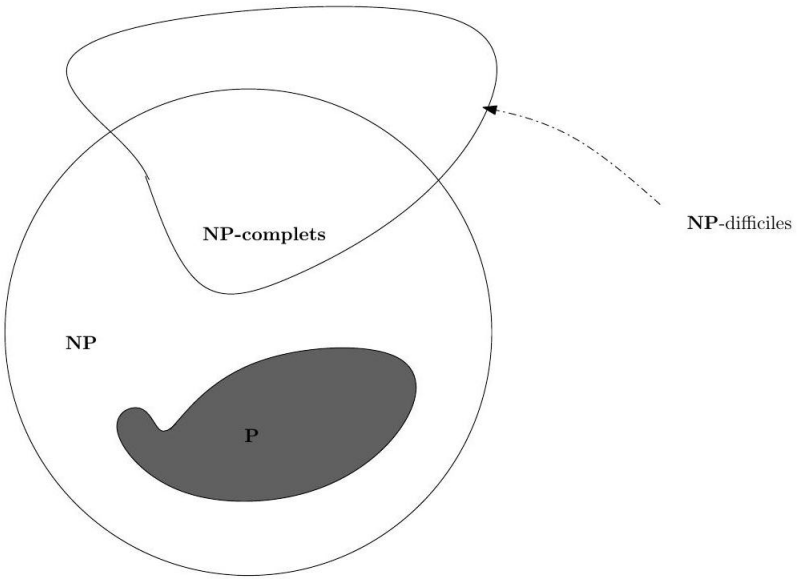


FIGURE 3.4 – Les problèmes NP

Bibliographie

[1] Gabrielle Demange and David Gale. The strategy structure of two-sided matching markets. *Econometrica : Journal of the Econometric Society*, pages 873–888, 1985.

[2] Dumont Dominique. Mariages stables. *Séminaire lotharigien de combinatoire*, pages 1–12, 1990.

[3] Lester E Dubins and David A Freedman. Machiavelli and the gale-shapley algorithm. *The American Mathematical Monthly*, 88(7) :485–494, 1981.

[4] Meunier Frédéric. Introduction à la recherche opérationnelle. *Université Paris Est*, pages 1–177, 2012.

[5] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American mathematical monthly*, 69(1) :9–15, 1962.

[6] Donald Ervin Knuth. Marriages stables. *Technical report*, 1976.

[7] Rigo Michel. Théorie des graphes. *Département de Mathématqiues*, pages 1–176, 2010.

[8] Murthy P. Rama. *Operations research*. New Age International, 2007.